Towards Modular and Certified Avionics for UAV

F. Boniol, V. Wiels
(Onera)

E-mail: frederic.boniol@onera.fr

DOI : 10.12762/2014.AL08-02

This paper proposes a review of the current state and forthcoming evolutions for UAV avionics architecture and software. It provides an outlook of the specific technical issues arising in the design of embedded systems for UAV.

Introduction

The Unmanned Aerial Vehicle (UAV) industry has been rapidly growing over the last decade. New UAV are being developed for military applications and also for civil usage. There is a strong correlation between the mission of a UAV and the avionics necessary to implement it. For this reason, the design, development and verification of UAV avionics, including hardware and software architecture, have been the subject of considerable research [23, 21, 16, 22].

UAV avionics, like those of traditional aircraft, are in charge of implementing flight control and flight navigation. However, they should also ensure a desired level of autonomy and the control of the payload (if any).

For flight navigation, a UAV includes embedded means for estimating, at any time and anywhere, its position, speed and acceleration. This requires navigation sensors (such a GPS) and robust estimation algorithms. For flight control, the UAV must generate the steering commands and subsequent control surface deflections to stabilize the vehicle and to adequately follow the flight plan. This again requires robust control algorithms. These computations are relatively simple compared to the flight planning algorithms. They however require the use of accurate real-time processors and operating systems.

In addition to the flight control and navigation part, UAV require specific autonomy means. The autonomy requirement is the main difference between UAV and manned aircraft. Autonomy is the ability to operate without direct control from a ground operator. Complex or faraway missions without ground infrastructure (for instance data link means and ground stations) would necessitate making the UAV increasingly autonomous. Autonomy requires specific sensors, such as optical devices, and complex software, such as image processing software and intelligent flight planning. Ideally, the UAV must have the capability to plan and re-plan its own flight plan. This results in the requirement for an on-board high-performance computing architecture where flight-planning algorithms can be run. These algorithms require knowledge of the UAV's surroundings, including other traffic, weather, obstacles, fuel usage, flight time, etc. Furthermore, in the event of failure, the UAV must have the capability to reconfigure itself and re-plan its trajectory or its mission. These autonomy requirements result in complex software, which requires high performance computing means without compromising safety: efficient techniques are necessary to verify and validate software.

The aim of this article is to discuss new challenges for future UAV avionics architectures and software : current state and forthcoming evolutions of UAV avionics, use of IMA (Integrated Modular Avionics) for UAV and certification issues.

Current state and forthcoming changes

Current state

The main challenge encountered by UAV avionics is to safely operate on-board two types of computation: flight control/navigation and flight planning/re-planning, including the reconfiguration of the avionics itself in case of mission re-planning.

In order to respond to this challenge, the first generation of UAV avionics architectures were divided into three loosely coupled physical parts. The first one is dedicated to navigation and flight control; the second one offers sensors, hardware and software components ensuring the desired level of autonomy; while the third part controls the payload of the UAV. The second and third parts are generally specific to the operational role that the UAV is supposed to carry out. In most cases, each part is implemented by a monolithic dedicated platform composed of the simplest possible processor with its own resources (memory and communication bus) (figure 1). UAV developed in the 90 s and 2000 s were based on this principle (see for instance appendix A of [27], and the Piccolo architecture in [33]).

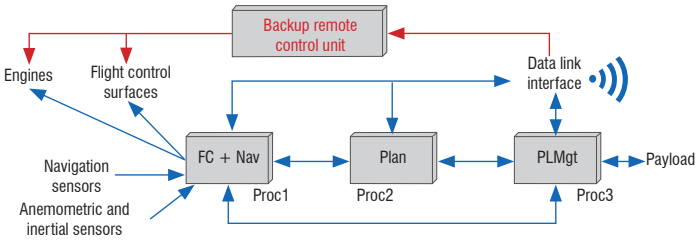

Figure 1 – Typical UAV 1st-generation architecture

Such a physical segregation between the three parts ensures that they operate (nearly) independently from each other. More precisely, the autonomy sensors and algorithms do not interfere with the control loop and vice-versa. Likewise, the payload software does not affect the rest of the avionics. Thus, a failure in the payload part should not affect the safety of the vehicle. Thus, each part can be designed, developed, dimensioned and certified separately, without considering interferences coming from the other parts. Finally, in the event of failure in one of the first two parts, a human operator can directly control the vehicle from the ground by means of a set of data-links and specific components. The vehicle becomes in that case a remote-controlled plane.

However, these first architectures based on the principle of separation of concerns have many limitations.

Forthcoming changes

Firstly, safety is guaranteed in the final degraded mode (i.e., the mode in which the automatic flight control part is lost) by the ability to pilot the vehicle from the ground. As stated above, it assumes (1) safe onboard mechanisms to commute from the automatic flight mode to the remote-controlled flight, (2) a data-link between the vehicle and the group operator, which guarantees that orders and data are transmitted in real-time (in less than 10 or 100 milliseconds for flight control orders) and, (3) a ground infrastructure able to present to the human operator the complete situation of the vehicle (position, speed, attitude, obstacles in front of the vehicle, etc.). Such requirements are not consistent with complex and faraway missions, or with missions in a hostile environment. In that case, contrarily to manned aircraft, the UAV must ensure its own safety without waiting for backup orders sent by a human pilot. However, first generation UAV avionics architectures do not offer the appropriate safety level. This is their first limitation.

Secondly, the continual development of UAV applications results in an ever-increasing demand on embedded algorithms. On board computational resources must meet this demand, while at the same time providing robustness, reliability and a small footprint, both in physical size, mass and power consumption.

Thirdly, new applications may necessitate the integration in a more coupled way of the three parts of the UAV avionics. In particular, the payload management may depend on flight data, such as position, speed, attitude, etc. Conversely, navigation and flight control may depend on the state of the payload. This requires an appropriate mechanism providing navigation data to the payload and conversely, in such a manner that the payload activity cannot interfere with flight control and planning. Failure of the payload must not compromise the safety of the UAV (for example, denying access to the on-board data-bus by saturating it with payload messages). First generation UAV avionics architectures do not offer such a mechanism.

A first solution could be to continue with the segregation principle (each part has its own sub-architecture), while increasing the number of computing resources. It should lead to the duplication of several components (for instance, the flight data calculation for the payload). However, this first architecture principle reaches its natural limit when the weight and volume of the dedicated sub-architectures encounter the envelope restrictions of the UAV. This issue becomes central in the case of small UAV able to carry only a few kilograms (generally less than 10 kg) including payload and avionics. Another drawback becomes obvious: the huge number of different resources has significantly increased the maintenance costs in terms of component spare part provisioning and handling.

Another approach, called Integrated Modular Avionics (IMA) [2, 3] has been suggested to address this issue for manned aircraft, such as Airbus A380 / A350 and Boeing B787.

Towards modular integrated avionics for UAV

Modular Integrated Avionics

Resource sharing and robust partitioning are the central ideas of the IMA concept. They are based on two principles: partitioning principles in processing modules and partitioning principles for communications between functions.

Processing module partitioning

As has already been explained, an UAV avionics architecture implements several software functions (flight control, navigation, planning and payload management), each of them possibly divided into sub-functions. Initially running on different processors, the first IMA idea is to place these functions on processing modules partitioned with respect to space (resource partitioning) and time (temporal partitioning).

- *Resource partitioning.* A processing module is divided into partitions. Each partition is seen as a virtual processing module. It is allocated a set of private spatial resources (memory, non-volatile memory, I/O resources, etc.) in a static manner. Low-level mechanisms (at the operating system level) provide protection for partition data against any modification from the other partitions. They monitor function activity with reference to allowed resources, which are statically allocated through configuration tables.
- *Temporal partitioning.* Each function is allocated a partition. The scheduling of partitions on each module is defined off-line by a periodic sequence of slots, statically organized in a time-frame. Each partition is allocated a time slot for execution. At the end of this time slot, the partition is suspended and execution is given to the next partition (running another function). Thus, each function is periodically executed at fixed times.

Thanks to partitioning mechanisms, functions become independent. A faulty function can be isolated without affecting functions placed on the same module.

Communication resource partitioning

Initially routed onto different physical links, the second IMA idea is to place communications between functions on shared communication networks. The network is divided into Virtual Links (VL). Each VL is dedicated to the traffic coming from a single function. It is characterized

by a bounded bandwidth. Similarly to processing modules, a low-level mechanism (at the network level) guarantees that no function can go beyond its contract, that is, produce more communication than the permitted bandwidth. Such a mechanism can be implemented by a traffic shaper, which separates two successive emissions on the VL by at least a fixed time interval called Bandwidth Allocation Gap (BAG). This principle has been implemented in the Avionics Full Duplex Ethernet (AFDX) architecture embedded in the Airbus A380 and A350 [3].

A typical IMA platform is described in figure 2. Its hardware architecture consists of 3 generic computing processing modules (called CPM) that are connected to a communication network. The network is composed of two identical redundant parts (Part A in blue, and Part B in red). CPM1 and 2 are connected to Switches 1 (A and B), while CPM3 is connected to Switches 2 (A and B). Flight control and navigation sensors and actuators are reached through a redundant gateway (Gtw1) connected to Switches 1. Similarly, the payload and the data link interface are reached through a second redundant gateway (Gtw2). As shown in the figure, the critical functions Flight Control (FC) and Navigation (Nav) are triplicated and the Planning Function (Plan) is duplicated, while the Payload Management Function (PLMgt) is implemented by a single occurrence (*i.e.*, without any redundancy). Each CPM is divided into four partitions (*e.g.*, FC1 is hosted in the first partition of CPM1). On each module, partitioning and scheduling are ensured by a partition manager, while bandwidth communication from each function is controlled by a VLs manager. Figure 3 shows the time-triggered scheduling of the four partitions hosted by CPM1. As has already been explained, this scheduling is organized as a sequence of time slots. It is composed of two minor frames (MiF), the duration of which is 10 ms. FC1 runs in the first time slot of each MiF. The duration of this time slot is 2ms. FC1 is then supposed to execute every 10 ms within an execution time of less than 2 ms. Nav1 runs only in the first MiF, while PLMgt runs in the second MiF. The aim of the partition manager is to unroll this sequence and monitor each partition. For instance, if PLMgt tries to continue after the end of its time slot, the partition manager stops it and starts the next partition (FC1). Hence, a software failure in PLMgt does not affect FC1. Note that in the CPM1 scheduling, a spare time slot is reserved for hosting potential new functions without affecting other functions.

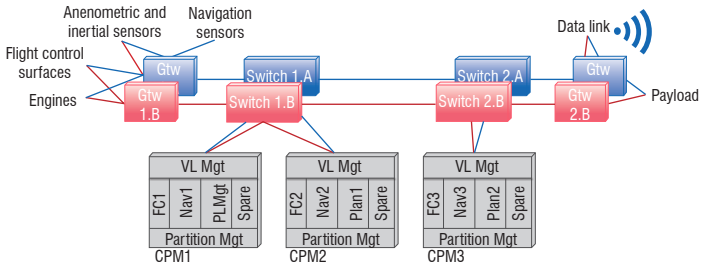

Figure 2 – Example of UAS IMA architecture

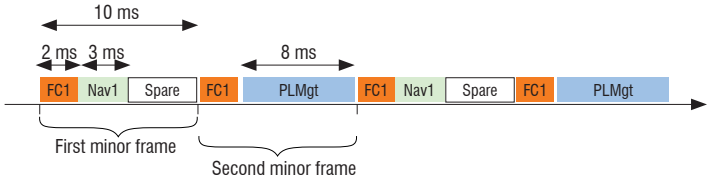

Figure 3 – Temporal scheduling of CPM1

Benefits and effects for UAV avionics

The benefits of such a new architecture are mainly: safety level improvement, as well as weight and power consumption reduction. Let us again consider the architectures given in figure 1 and figure 2. These two architectures implement the same functions FC, Nav, Plan and PLMgt. They are both composed of 3 processors. However, the first architecture is not fault tolerant. For instance, loss of Processor 1 leads to total loss of the vehicle. Conversely, the IMA architecture (figure 2) is fault tolerant with the same number of processors. For instance, despite of the loss of CPM1, the flight control and navigation functions still run properly on CPM2 and CPM3. Only the payload management is lost. It is obvious that the total loss of the vehicle is consecutive to at least two failures: for instance, loss of Switches 1.A and 1.B or loss of Gateways 1.A and 1.B, etc. In that sense, with (nearly) the same number of resources, the IMA architecture is safer than the first one.

Globally, IMA results in a reduction of the required physical resources. Reduced physical resources translate into global weight and power savings for the UAV. The same trend has been observed in aircraft architectures: for instance, the number of processing units in the A380 is half that of previous generations. Reductions in operating costs are expected to be significant, with the decrease in the number of computers and cables (for power supply or communication), contributing to a reduction of vehicle weight leading to better fuel consumption efficiency and then to a greater autonomy.

Past and recent experiments on applying IMA to UAV

Considering these expected benefits, recent research has been conducted on the integration of IMA architecture into modern UAV. A preliminary work has been proposed by Elston *et al.* [17]. They are developing a distributed modular architecture concept for small UAV (about 10 kg). This architecture is composed of a set of computing modules communicating through a CAN bus. Similarly, Ellen *et al.* investigated in [16] an architecture for the QUT research UAV, still based on a cluster of small dedicated processors communicating through CAN buses. They show that the performances of this architecture, in terms of power consumption, size and weight, are better than those for the legacy architecture (based on a centralized PC104 computer). However, contrary to the full IMA concept, computing modules in these proposed architectures still own their private sensors and actuators and host only one function. There is no partitioning mechanism.

Following this direction, Lopez *et al.* investigated in [25] a middleware-based architecture suitable to operate as a flexible payload and mission controller in a UAV. The architecture is composed of low-cost computing devices connected by a network. The functionality is divided into reusable services distributed over a number of nodes, with a middleware partitioning their lifecycle and communication. However, the middleware does not take into account real-time issues. Thus, flight control and navigation cannot run on this platform and still require a dedicated real-time architecture.

In order to respond to the real-time issue, [29] proposes an architecture platform based on a Time-Triggered network. Functions, including flight control and navigation, run on dedicated PC/104 computers and communicate in a deterministic way through the network. Thanks to the time-triggered protocol, the network guarantees fixed time slots for each function. This solution has been implemented on large UAV, such as the R-MAX Helicopter (about 10 kg).

More recently, [22] has developed a complete IMA solution, based on an ARINC 653 processing module, for a small quad-rotor helicopter. Due to the UAV size (70 cm in diameter and 1 kg payload), the avionics only include a single embedded processor hosting the flight control, navigation and planning functions. The processing unit is partitioned in a deterministic way according to the IMA principle. To our knowledge, this experiment is the first attempt to apply the IMA concept, here reduced to the processing module part, to small UAV.

All of these experiments clearly show the trend to embed an IMA execution platform for flight control and the navigation function, as well as for planning and payload management, in both small and larger UAVs.

Certification issues

Given that the use of drones for different kinds of mission is spreading, and will continue to do so, their safety and security will become a crucial issue. For aircraft, safety and security are assessed using certification standards. The ARP 4754 [4] is the standard for systems, DO-178 [14] is the standard for software, while Common Criteria (ISO 15408) [12] handles security aspects.

These standards are bound to also be applied for UAV systems and software. Application of ISO 15408 will enforce security requirements and help to prevent the hacking of UAV. However, it will only be effective if safety-critical requirements are also taken into account. Ensuring the safety of embedded software is paramount, since there is no human pilot onboard. In this section, we consider the challenges at stake for the certification of this software. We distinguish the domains where aircraft solutions can be adapted to UAV without too much trouble and UAV specific certification issues.

[28] gives a broader overview of the challenges and a roadmap for the certification of Remotely-Piloted Aircraft Systems. We focus on software aspects, but also consider autonomous systems (even though their operational use is further away). [28] also tackles insertion into the airspace; we do not consider this certification issue.

Issues similar to those for aircraft

Flight control and navigation systems are certified for aircraft, using classical means or more recently innovative verification techniques. We will not provide an exhaustive overview of existing work here, but rather only list the various aspects that should be considered together with a few references, mainly of Onera work in this domain. We focus on aspects related to avionics; safety and security assessment at the aircraft and system levels are also essential, but will not be discussed here (see [7,5]).

Real-time analysis

Certification objectives regarding real time are scattered around in various certification standards (IMA, software) but they are essential for the correctness of software and systems. Regarding real-time behaviors, the first requirement is to guarantee that each function located in an IMA partition terminates properly before the end of the partition. For instance, let us consider the flight control function FC1 hosted by CPM1 (figure 2). FC1 runs in a partition of which the duration is 2 ms (figure 3). Thus, it must be shown that all sub-functions involved in FC1 are scheduled in such a way that they will all terminate before 2 ms have elapsed. Several techniques and tools have been developed to analyze worst-case execution time [1, 26], and worst-case response time [30, 11] for IMA software, or generate a correct scheduling from different constraints within a partition [30].

The second certification requirement deals with worst-case traversal time through a communication network. Let us again consider the example in figure 2. FC1 periodically sends orders to actuators through the network. Note that the payload also sends and receives data through the same network. It could then happen that, if a failure occurs somewhere in the payload, it may begin to send a huge amount of data to PLMgt, overloading the communication network, leading to delays in the flight control orders. Such a scenario may lead to a catastrophic situation, despite the initial single failure being of minor importance. An interesting benefit of the IMA principle is that, if functions are statically allocated in modules and partitions, and if the network hosts mechanisms enforcing functions to respect their communication contract (*e.g.*, traffic shapers), then it is possible to mathematically prove that the end-to-end delay of any message is bounded and it is possible to evaluate an over-approximation of this bound. This proof is based on the network calculus theory [8, 9]. Network calculus has been used for certification of the A380 and A350 avionics network. It contributes an adequate mathematical technique for UAV avionics network certification as well.

Software verification

DO-178/ED-12 [14] does not prescribe a specific development process for software, but rather identifies important activities and design considerations throughout a development process and defines objectives for each of these. DO-178 [14] distinguishes development processes from “integral” processes that are meant to ensure correctness, control and confidence in the software life cycle processes and their outputs. The verification process is part of the integral processes, along with configuration management and quality assurance. Version C of this standard, which was published in 2011, includes technical supplements to take into account and facilitate the appropriate use of new software engineering techniques. DO-333/ED-216 [15] is the formal method supplement. Formal methods can be applied to many of the development and verification activities required for software. The supplement proposes guidance for the use of formal methods. It describes the activities that are needed when using formal methods, new or modified objectives and the evidence needed for meeting those objectives.

Formal verification techniques have already been used for the certification of aircraft avionics software [32] and a lot of work is underway in this field [34]. Specific work on the verification of stability and safety properties of flight control software could be of special interest for UAV [10].

UAV specific issues

In this section, we point out the specific certification issues arising for UAV in the various domains considered previously.

In-flight reconfiguration

As explained above, IMA architectures are based on a strict principle: static and fixed allocations. However, it could be interesting, in the event of a hardware failure or in the event of loss of the communication link, for example, to be able to reconfigure the system, which means reallocating

functions to safe processors. Let us consider the example in figure 2 and let us imagine that CPM1 fails. Then PLMgt is lost. It could be interesting to reallocate it in the spare partition of CPM2. Such a mechanism could allow a reduction of the number of on-board processors, thereby saving weight, particularly for small UAV. Unfortunately, current IMA architectures do not allow in-flight reconfiguration. Recent research work conducted by Onera with Thales and Airbus has explored the reconfiguration issue for aircraft IMA architectures in the European SCARLETT project (http://www.scarlettproject.eu/) [6]. The solution is limited to on-the-ground reconfigurations, which seems to be enough for aircraft architectures. However, small UAV can only include a small number of embedded resources. Safe in-flight reconfiguration remains a strong challenge for UAV architectures.

Software verification

A significant difference between aircraft and UAS resides in mission management software. As stated by [23], mission management software may be quite complex, in order to be able to respond to various situations; it may include various concurrent tasks, etc. Moreover, the development of mission management software typically does not follow stringent processes, such as those used for flight critical software; the verification of this software is currently mostly done through simulations and flight tests. The proliferation of UAV will call for the use of more rigorous means of verification for mission management software.

Requirements for this kind of software will first have to be identified and formalized. It may not be an easy task, due to the very nature of the software. In order to ensure the autonomy of the UAV, mission management software is designed to be “intelligent”, to be able to respond to many different situations by analyzing available information. An exhaustive enumeration of all possible situations might be a tedious and difficult task. Once the requirements have been expressed, formal verification techniques will also have to be adapted, or extended, to handle the specificities of mission software. A family of techniques that could be useful for the verification of mission software is runtime verification. The principle is to monitor the software with respect to a given set of formalized properties [18, 20].

Conclusion

In this paper, we have described the current state of avionics for UAV, identified challenges in this domain and proposed directions for future work. In conclusion, we would also like to mention an Onera initiative, called FORC3ES (Formal engineering for certified control-command embedded systems). This initiative is aimed at defining a set of techniques and tools for the formal development and verification of control-command systems. The framework is experimented with on a UAV and its associated Iron Bird (an Iron Bird is a system test bench; it includes the same sensors, actuators and avionics as the real aircraft; see pictures in figure 4). The first part of this project is dedicated to flight control software development and verification, but in the long run we also intend to study the verification of mission management software and to experiment with new concepts of IMA architectures for UAV ■

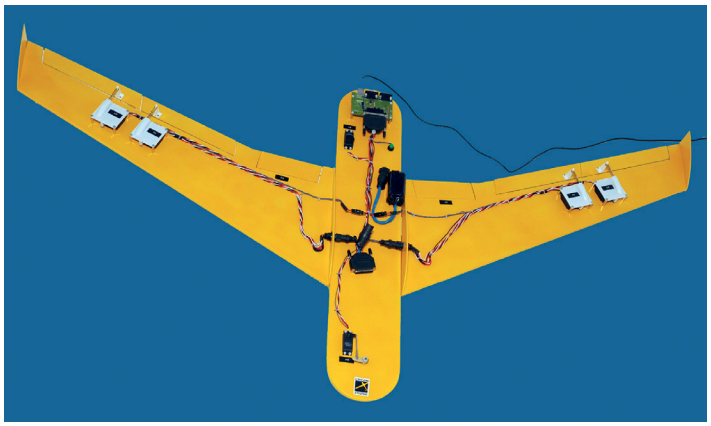

Figure 4 – UAV (left) and associated Iron Bird (right)

References

[1] *aiT Worst-Case Execution Time Analyzers*. http://www.absint.com/ait.
[2] Aeronautical Radio Inc – *ARINC 653: Avionics Application Software Standard Interface*. 1997.
[3] Aeronautical Radio Inc – *ARINC 664: Aircraft Data Network, Part 1: Systems Concepts and Overview*. 2002.
[4] ARP64754A – *Guidelines for Development of Civil Aircraft and Systems*. 2010.
[5] P. BIEBER, J.-P. BLANQUART, G. DESCARGUES, M. DULUCQ, Y. FOURASTIER, E. HAZANE, M. JULIEN, L. LÉONARDON and G. SAROUILLE – *Safety and Security Assurance in Aerospace Embedded Systems*. ERTSS, Toulouse, February 2012.
[6] P. BIEBER, E. NOULARD, C. PAGETTI, T. PLANCHE, F. VIALARD – *Preliminary Design of Future Reconfigurable IMA Platforms.* SIGBED Review 6(3): 7, 2009.
[7] P. BIEBER and C. SEGUIN – *Safety Analysis of the Embedded Systems with the AltaRica Approach*. Industrial Use of Formal Methods: Formal Verification, Wiley, 2012.
[8] J.-Y. LE BOUDEC and P. THIRAN – *Network Calculus: a Theory of Deterministic Queuing Systems for the Internet*. LNCS, Vol. 2050, Springer, 2001.
[9] M. BOYER, N. NAVET, X. OLIVE and E. THIERRY – *The Pegase Project: Precise and Scalable Temporal Analysis for Aerospace Communication Systems with Network Calculus*. LNCS, Vol. 6415, pp. 122–136, Springer, 2010.
[10] A. CHAMPION, R. DELMAS, M. DIERKES, P.-L. GAROCHE, R. JOBREDEAUX and P. ROUX – *Formal Methods for the Analysis of Critical Control Systems Models: Combining Non-linear and Linear Analyses*. FMICS, pp. 1-16, 2013.
[11] Cheddar, Université de Brest, http://beru.univ-brest.fr/~singhoff/cheddar.
[12] Common Criteria, http://www.commoncriteriaportal.org.
[13] R.P.G. COLLINSON – *Introduction to Avionics Systems.* http://books.google.fr/books?id=aU8SMhzrScgC, Springer, 2011.
[14] RTCA/DO-178C, EUROCAE/ED-12C – *Software Considerations in Airborne Systems and Equipment Certification.* 2011.
[15] RTCA/DO-333, EUROCAE/ED-216 – *Formal Methods Supplement to DO-178C and DO-278A.* 2011.
[16] R. ELLEN, P. ROBERTS and D. GREER – *An investigation into the Next Generation Avionics Architecture for the QUT UAV Project.* Goh, Roland & Ward, Nick (Eds.) Smart Systems 2005 Postgraduate Research Conference, Brisbane,15 December, 2005.
[17] J. ELSTON, B. ARGROW and E. FREW – *A Distributed Avionics Package for Small UAVs.* AIAA Conference, Arlington, Viginia. 2005.
[18] Y. FALCONE, K. HAVELUND and G. REGER – *A Tutorial on Runtime Verification. Book chapter four: Summer School Marktoberdorf 2012 - Engineering Dependable Software Systems. July 31 to August 12, 2012.* IOS Press book, NATO Science for Peace and Security Series - D: Information and Communication Security, Vol. 34, 2013.
[19] V. GAVRILETS – *Avionics Systems Development for Small Unmanned Aircraft.* Master's Thesis, MSc in Aeronautics and Astronautics, MIT, 1998.
[20] A. GROCE, K. HAVELUND, G. HOLZMANN, R. JOSHI and R-G. XU – *Establishing Flight Software Reliability: Testing, Model Checking, Constraint-Solving, and Monitoring.* Annals of Mathematics and Artificial Intelligence, March 2014.
[21] G.Y. IMMANUEL and N. JOHNSON – *New Architectures for UAV Flight Control Avionics.* 21st Digital Avionics Systems Conference, 2002.
[22] H.C. JO, S. HAN, S.H. LEE and H.W. JIN – *Implementing Control and Mission Software of UAV by Exploiting Open Source Software-Based Arinc 653.* Digital Avionics Systems Conference (DASC), IEEE/AIAA 31st, pp. 8B2–1–8B2–9, October 2012.
[23] T. JOHNSON, R. KONECK and S.F. BUSH – *Improving UAV Mission Success Rate through Software Enabled Control Design.* IEEE Aerospace Conference, 2000.
[24] M. LAUER, J. ERMONT, F. BONIOL and C. PAGETTI – *Worst Case Temporal Consistency in Integrated Modular Avionics System.* HASE, T. M. KHOSHGOFTAAR Ed., IEEE Computer Society, pp. 212–219, 2011.
[25] J. LÓPEZ, P. ROYO, E. PASTOR, C. BARRADO and E. SANTAMARIA – *A Middleware Architecture for Unmanned Aircraft Avionics.* Middleware'07, Newport Beach, California, November 26-30 2007.
[26] OTAWA, IRIT Université de Toulouse, http://www.otawa.fr.
[27] S. PARK – *Avionics and Control System Development for Mid-Air Rendezvous of Two Unmanned Aerial Vehicles.* PhD Thesis, MIT, 2004.
[28] *Roadmap for the Integration of Civil Remotely-Piloted Aircraft Systems into the European Aviation System.* Final report from the European RPAS steering group, June 2013.
[29] A. SAMUEL, N. BROWN, R. COLGREN and G. KELLY – *Subsystem Design and Integration of A Robust Modular Avionics Suite for UAV Systems Using the Time Triggered Protocol (TTP).* SAE Aerospace Technology Conference, 2007.
[30] SchedMCore, Onera, http://sites.onera.fr/schedmcore.
[31] A. SHAHSAVAR – *Defining a Modular, High Speed and Robust Avionic Architecture for UAV's.* Master's Thesis Programmes in Engineering Space Engineering, 2008:174, Lulea University of Technology. 2008.
[32] J. SOUYRIS, V. WIELS, D. DELMAS and H. DELSENY – *Formal Verification of Avionics Software Products*. Formal Methods, 2009.
[33] B. VAGLIENTI and R. HOAG – *A Highly Integrated UAV Avionics System.* Technical Report, A cloud cap Technology, 2003.
[34] V. WIELS, R. DELMAS, D. DOOSE, P.-L. GAROCHE, J. CAZIN and G. DURRIEU – *Formal Verification of Critical Aerospace Software.* AerospaceLab, Issue 4, 2012.

Acronyms

AFDX	(Avionics Full Duplex Ethernet)
BAG	(Bandwidth Allocation Gap)
CPM	(Computing Processing Module)
FC	(Flight Control)
IMA	(Integrated Modular Avionics)
MiF	(Minor Frame)
PL Mgt	(Payload Management)
UAS	(Unnamed Aerial System)
UAV	(Unnamed Aerial Vehicle)
VL	(Virtual Link)

AUTHORS

Frédéric Boniol graduated from a French High School for Engineers in Aerospace Systems (Suapero) in 1987. He holds a PhD in computer science from University of Toulouse (1997). He has a research position at Onera/DTIM. His research interests include modeling languages and performance analysis methods for embedded real-time systems.

Virginie Wiels is research scientist at Onera/DTIM since 1998. She is working on formal verification of embedded systems and software. Before joining Onera, she was research associate at NASA/WVU IV&V Facility in Fairmont (USA). She holds a PhD in computer science from University of Toulouse (1997).