Unmanned Aerial Vehicles & Service-Oriented Architecture: LARISSA and Knowledge Based Framework's First Study

Emerson A. Marconato, Douglas Rodrigues,
Arthur A. Chaves, Kalinka R. L. J. C. Branco
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, Brazil
Email: {emerson, douglasr, kalinka}@icmc.usp.br
arthurac@usp.br

Rajiv Ramdhany, Geoff Coulson
School of Computing and Communications
Lancaster University
Lancaster, UK
Email: {r.ramdhany, g.coulson}@lancaster.ac.uk

***Abstract*—Embedded systems are computer systems that are part of a larger system, which generally provide real-time monitoring and control. They execute a predefined set of tasks on behalf of a real-time application, and may have special requirements based on the application domain they support. For instance, these systems are considered safety-critical embedded systems when failure may result in loss of life or high-value assets. UAVs (Unmanned Aerial Vehicles) constitute a typical application of a complex critical embedded system. One concept that can result in radically different solutions in UAVs is the use of Service-Oriented Architecture (SOA) based on standard reference model architecture. The increasing use of SOA in critical applications demands dependable and cost-effective techniques to ensure high security. In this paper we developed different kind of services for avionics with different parameters (security, reliability and performance) to provide use of SOA in a less critical part in the whole systems. Both LARISSA (Layered Architecture Model Interconnection Systems in UAV) and KBF (Knowledge Based Framework for Dynamically Changing Applications) presented in this paper can give more intelligence to UAVs and provide a new way of segregating the UAV mission from the vehicle itself. Some services were developed and a performance evaluation was conducted showing the benefits in choosing some determined services.**

***Keywords*—Critical embedded systems, UAVs, SOA, Web services, performance.**

I. Introduction

Embedded systems are computer systems that are part of a larger system. These systems provide, in most cases, real-time monitoring and control. They are considered safety-critical when possible failure may result in loss of life or high-value assets [1][2][3][4]. Both hardware and software in embedded systems have become increasingly complex. Multicore and multiprocessor systems have become common, which has further increased the complexity of software [5]. Moreover, both can be seen in homes and in business environments where they have been used for the control or management information.

UAV is a typical application of a critical embedded system. The term UAV was adopted by FAA (Federal Aviation Administration) and by the international academic community to describe a system that includes not only the aircraft, but also all the associated elements such as payload, ground control station and communication links [6]. UAVs have been widely used in precision agriculture, national security and environmental monitoring. Several papers have been published in this area, demonstrating the feasibility of using such vehicles as important tools for performing precision agriculture and environmental monitoring [7][8].

There are different types of UAVs that have different capabilities. Some aircraft can fly autonomously, following a pre-programed flight path (based on a grid or a sequence of waypoints) [7], while others can fly receiving commands from ground stations operated by pilots. The aircraft's size can range from micro to large, and the ground control station can be implemented on smartphones, tablets, laptops or networks of workstations (distributed control stations). Thus, the aircraft may vary not only in size, but also in shape and type of propulsion performance. The ground station interface can vary from a joystick to a tangible user interface (for example, a table with tangible augmented reality). The performance of the communication links and the payload type are also very important to fulfil the mission intended for the system. Specialized literature says that UAVs will become popular and will be part of airspace in the next 10 years, performing different missions, from agricultural border inspections to automatic cargo transport [9][10][11].

The UAVs' heterogeneity and constraints and the distinct nature of their interactions are challenges for their successful integration into architecture for a shared exploitation of UASs. The heterogeneity prevalent in UAVs in terms of services for avionics and architecture is particularly relevant to elaboration of multi-application missions. This heterogeneity in UAV services is often manifested as characteristics, such as reliability, security, and performance. Different service implementations typically offer different guarantees in terms of these characteristics and in terms of associated costs. The initial choice of a particular avionics service implementation can therefore become sub-optimal as long new applications/services are deployed, needing a careful selection of services to fulfil particular performance and operational guarantees and, subsequently, to avoid compromising the mission.

In the same way, architectures that enable the organization and more specific definition of the components of these embedded systems (UAVs) ease the development of hardware and software that compose them, allowing these vehicles be more easily inserted and incorporated in a non-segregate airspace. Therefore, the main goal of this preliminary research is propose a new architecture to UASs and investigate the degree of heterogeneity present in UAVs in terms of services, proposing architectural abstractions for the integration of these service variants. In particular, we explored the notion of Service-Oriented Architecture (SOA) in the context of UAVs as safety-critical embedded systems for the composition of services to fulfil the specified application performance and the dependability guarantees.

The rest of this paper is organized as follows. In Section II, we describe the related works in the field of SOA and Reference Model Architecture in embedded and safety-critical embedded systems. Section III shows the concepts of Reference Model Architecture and the new trends using SOA in this kind of system. Section IV presents KBF (Knowledge Based Framework for Dynamically Changing Applications) and its functioning. Section V shows a case study of default, secure, and reliable services, and then analyses the results. Finally, Section VI presents our conclusions regarding this work, as well some future prospects for this research.

II. Related Work

This section presents a review of Reference Model Architecture and SOA (Service-Oriented Architecture) in embedded systems and critical embedded systems. The Reference Model Architecture found in the literature can be classified in Federated (traditional architectures) and Non-Federated (non-conventional architectures).

NIST (National Institute of Standards and Technologies) provides a reference model for UAVs [12]. In this particular pattern, the reference model was proposed to specify military rules, practices and controls in a comprehensible and intuitive way for a human commander. The proposed architecture approach is different from that required in our project, focusing on a lower level of abstraction, which the layers specify what components found in a UAV system should do. Then the authors introduced a hardware/software embedded architecture especially designed to operate as a UAV's payload and mission controller. The hardware architecture is built as a set of embedded microprocessors connected by a LAN (Local Area Network). Over this hardware infrastructure is implemented a software layer that allows each module to support multiple applications. Every application can create and sign services and these services can be dynamically discovered and consumed as in Internet domain.

The work proposed by Pastor et al. [13] is based on architecture of hardware and software designed to operate the mission controller and the payload in mini/micro UAVs. According to the authors, the innovation is the use of a distributed hardware architecture which is easily scalable by the use of LAN architecture-based software subscription services, communication abstraction layer and execution flow based on mission planning. Still according to the authors, the high level of modularity offered by a LAN provides flexibility for coupling the microprocessor type most appropriate to use the module, given its functional requirements.

Olson et al. [14] proposed another architecture model. This is Phase III of the project named MCAP (Manned/Unmanned Common Architecture Program), used by the U.S. Army in UAVs such as FCS (Future Combat Systems) and C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance). Phase III of MCAP architecture is based on electronics and commercial off the shelf (COTS) and open standards interfaces. The objective of the development of the model was define and develop an architecture capable of supporting a amount of UAVs in the U.S. Army platforms, demonstrating the performance of the resulting system in a laboratory environment. The development of this model relied on the study of three classes of UAS: Unmanned Combat Armed Rotorcraft (UCAR), an unmanned combat helicopter; Class IV Medium Altitude Long Endurance (MALE); and Extended Range/Multi-Purpose (ERMP), with two aircrafts: Fire Scout and Shadow 200.

The project presented by Neto et al. [15] is a modular embedded architecture, consisting of three levels: embedded systems, communication link, and inertial navigation system. The project's purposes are design and build a platform for research and development of UAV with autonomous behaviour. The proposed architecture consists of modular embedded electronics and communication protocols based on the OSI model.

Prisaznuk [16] proposed IMA, an integrated model of avionics that is used as architecture for conventional aircraft. IMA was initially proposed to be used in commercial and military aviation. It is set around the concept of high computational processing power and OS modules that allows independent processing of the application processing software. The modules share hardware resources and are allocated in offices, which have well-defined interfaces with the aircraft.

According to Watkins and Walter [17], it is possible differentiate IMA (Integrated Modular Avionics) architecture from the federated architecture (conventional). The authors state that the IMA architectures provide sharing of processors when processing information, communications and I/O. The resource row is divided for use of multiple avionics functions. The avionics functions served by IMA can be from different companies and their criticality is still guaranteed due to the robust partitioning mechanisms that are inherent in the architecture. In contrast, federated avionics architectures implement independent collections of dedicated computing resources (CPU, communications and I/O) for each avionics function normally contained in Line Replaceable Units (LRUs) or Line Replaceable Modules (LRMs). According to [18], other characteristics of IMA are the use of open standards and provision of a single data bus to interconnect the major aircraft systems.

Advantages of IMA compared to federated architecture are the economy of space, weight and power consumption, due a single unit performs various functions. Another advantage is the consolidation of hardware; it has several applications running on fewer processors [18].

Many complex embedded systems are coupled to a high-level information system. SOA can provide the integration of low-level embedded system services and high-level information system services. This integration is still an incomplete work, despite the many related works found in literature [19]-[25]. In practice, the use of SOA in embedded systems can provide a lot of benefits, such as decoupling configuration from environment, improvement of reusability and maintainability, higher level of abstraction and interoperability, more interactive interface between devices and information systems, and easy use of resource-hungry services provided by more powerful internet servers.

Using SOA and a reference model architecture is possible to get new improvements in critical embedded system. It is possible to take the advantages of the flexibility and can facilitate the modularization of the system components. It is also considered that the adoption and maintenance of standardized interfaces for UAVs can protect clients' investment in the development of new systems.

III. LARISSA: Layered Architecture Model Interconnection Systems in UAV

Architecture is a structure that identifies, defines, and organizes components. The relationship and the principles of design of components, functions and interface established between subsystems can also be defined by architecture. Moreover, a reference model for architecture is an architecture which the entities, relationships and information units involved in the interactions between and within the subsystems and the components are defined and modelled. In short, it is a model that incorporates the basic purpose and the idea of the system and can be considered as a reference for various purposes. The term architecture model used in this study reflects exactly that last statement: it incorporates the basic goal and ideas of the system.

The increasing use of UAVs should cause them to become increasingly common. In this scenario, the techniques proposed in this work will facilitate the development of automated applications for UAVs, allowing these vehicles be more easily inserted and incorporated into the airspace, contributing to their spread.

In order to propose a broader understanding of the component parts of a UAV system, we propose a layered model, which can be subdivided as needed. Figure 1 illustrates the model named LARISSA (Layered Architecture Model Interconnection Systems in UAV).

In LARISSA model, the components of a UAV may be divided into aerial segment and ground segment. The aerial segment is hierarchically composed of: (i) physical layer, (ii) distributed RTOS (Real Time Operating System) layer, (iii) system abstraction layer, (iv) monitoring and control layer, (v) navigation & services layer, and (vi) mission layer. The ground segment is divided into: (i) physical layer and (ii) ground station layer.

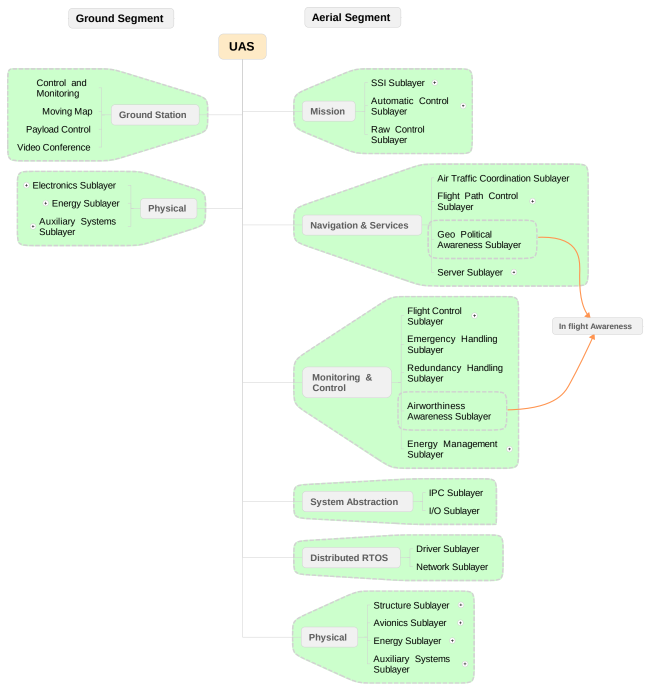

Fig. 1: LARISSA – The proposed reference model architecture.

The separation into layers allows the system to be divided into subsystems that can have different implementations and assists the separation of parts that compose a complex critical embedded system in different levels of criticality. Thus, the advantages offered by the service-oriented architecture can be applied in sections of low criticality, making the development of these sections simpler and more flexible.

These layers can be represented by models, which are intended to serve as guides for development of UAV systems, specifying how it will interconnect the various components, such as sensors, control circuits, GPS, payload, sensors, communication with the ground control station, and others.

In information technology, a layered architecture is used to define the specific responsibilities of each layer and the interconnection among them. Based on an architectural model, the hardware manufacturer or the software designer can develop their products knowing exactly which layer will interact in UAVs, which are the input and output parameters, and what type of connection must be used.

According to Tanenbaum [26], certain principles must be applied to define the layers: a layer must be created where there is need for other level of abstraction; each layer must perform a well-defined function, which should be chosen aiming the definition of standard protocols; layer limits must be chosen to reduce the flow of information transported among the interfaces; the number of layers should be large enough, so that different functions do not need be placed unnecessarily in the same layer, and small enough, so that the architecture does not become difficult to control. Architecture to be considered complete should define what each layer can perform, specifying services and protocols that are used in each one.

Moreover, the papers related to UAVs in the literature show UAVs implemented using traditional approaches [27][10][11]. On the other hand, there are roadmaps published periodically by military organizations (e.g., United States Air Force) illustrating the progress expected for UAVs, and they mentioned that in the future they may adopt an open, standardized, and scalable architecture, allowing the fast addition of modular functionality.

Each layer is composed of sub-layers, which are described in the next subsections. The navigation & services and mission sub-layers will be described in details because they are very important to the development of the framework proposed in Section IV.

A. *Aerial Segment's Physical*

The aerial segment's physical layer is the aircraft's hardware layer, which is decomposed in the structure, avionics, energy, and auxiliary systems sub-layer. Each sub-layer may be subdivided into more specific sub-layers.

B. *Distributed RTOS*

The distributed RTOS layer describes a set of API used by the real-time operating system embedded in the aircraft, used as input to or an output of the RTOS. In the driver sub-layer are the hardware drivers APIs, and in the network sub-layer are the network APIs.

C. *System Abstraction*

The system abstraction layer's function is defining a set of hardware for use in the upper layers. The IPC (Inter-Process Communication) sub-layer is responsible for the abstraction of communication among processes, and the I/O sub-layer controls the operation of input and output devices.

D. *Monitoring & Control*

The monitoring & control layer is responsible for monitoring the aircraft's actions, as well its control.

It is divided into the flight control sub-layer, emergency handling sub-layer, redundancy handling sub-layer, airworthiness awareness sub-layer, and energy management sub-layer. The flight control sub-layer responds to basic commands being executed by the aircraft, by the automatic take-off and also by the automatic landing. On the other hand, the emergency handling sub-layer is responsible for events that are not planned, such as battery consumption, making it impossible to accomplish the mission. The redundancy handling sub-layer manages duplicated subsystems in the aircraft that were installed to increase the reliance. The airworthiness awareness sub-layer is responsible for sensors and embedded detectors in aircraft, which purpose is to obtain information like those a human being has the capability to identify, such as smoke on board. The energy management sub-layer is responsible for the monitoring of energy levels consumed by the aircraft.

E. Navigation & Services

The navigation & services layer, illustrated in Figure 2, consists of the air traffic coordination, flight path control, geo political awareness, and server sub-layers. This layer is responsible for the aircraft's navigation, sending signals that perform the required path to accomplish the mission. The air traffic coordination sub-layer responds to traffic in the airspace in which the aircraft is operating. The flight path control sub-layer guides the aircraft's navigation to achieve the waypoints or the grid coordinates defined by the mission. The geo political awareness sub-layer is responsible for the virtual threshold that the aircraft must operate. The server sub-layer contains non-priority services that help navigation and mission accomplishment. These services can be based on WWW (World Wide Web) or DTM (Data Transfer Mechanism).

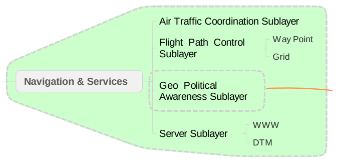
Navigation & Services
Air Traffic Coordination Sublayer
Flight Path Control Sublayer
Way Point
Grid
Geo Political Awareness Sublayer
Server Sublayer
WWW
DTM

Fig. 2: Navigation & services layer.

F. Mission

The mission layer, illustrated in Figure 3, is divided into SSI (Smart Sensor Interface), Automatic Control, and raw control sub-layers.

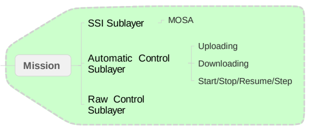

Fig. 3: Mission layer.

The SSI sub-layer is responsible for accessing the MOSA (Mission Oriented Sensor Array) [28] and performing the entire checking, allowing discovering whether the aircraft met all the attributes to accomplish the defined mission. The automatic control sub-layer is responsible for accepting the mission data (uploading), sending the collected data (downloading), and starting, stopping, resuming or performing part of the mission (start/stop/resume/step). The raw control sub-layer is simply responsible for sending data that does not need a proper treatment. With this layer, we ensure that data reaches a pre-set destination.

This layer is the last one from the aerial segment. The ground segment's layers are described in the next subsections.

G. Ground Segment's Physical

The ground segment's physical layer resembles, in some aspects, the air segment's physical layer. The division into sub-layers is presented as follows: electronics, energy, and auxiliary systems.

H. Ground Station

The ground station layer has control and monitoring, moving map, payload control, and video conference sub-layers.

The control and monitoring sub-layer receives information in the form of aircraft telemetry and can also issue commands to guide the aircraft. The moving map sub-layer is responsible for the exchange of maps ability, submitting new maps to the aircraft when the initial mission is changed. The payload control sub-layer sends signals to aircraft in order to control the movement and operation of sensors, cameras and radars. The video conference sub-layer is responsible for exchange sound and image with other control stations.

IV. Knowledge Based Framework for Dynamically Changing Applications (KBF)

To make possible the development of KBF, this paper considers critical embedded systems can be divided into sections of low and high criticality, based on LARISSA.

KBF was proposed in [29] and [30] and is currently under development. It extends the capability of a SOA broker's service discovery, adding knowledge about the application domain. Thus, KBF will use context and monitoring information to select or compose dynamically the best service to perform a specific mission. This selection or composition will be based on a set of usage rules and selection criteria, such as reliability, security, and performance. KBF is illustrated in Figure 4 and can be seen in detail in [9].

KBF uses a knowledge database to store all information and selection criteria defined by the user and by the application. Another key issue is the assembly of reconfigurable matrix, a data structure that correlates the chosen service, its functionality and the selection criteria to mission procedures. This matrix can be: static, semi-static and dynamic, depending on its composition and system operation [9]. Using all available information in the reconfigurable matrix, KBF can either choose or compose the best service to run a mission defined by the user.

V. Case study: fast, reliable, and secure web services in UAVs

The definition and specification of basic services that KBF can use were done, including their features and input/output parameters. These services were implemented using the Java programming language. These services use UAV's basic information (e.g., maximum cruise altitude, cruise speed).

In these experiments, we implemented the services listed in Figure 5. Those services were then replicated, adding reliability through WS-ReliableMessaging (WS-RM) specification [31], which is responsible for guaranteeing that messages are really delivered. Then those

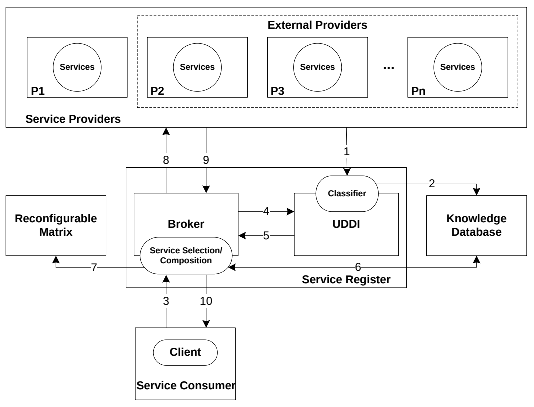

Fig. 4: Knowledge Based Framework for Dynamically Changing Applications (KBF)

services were replicated once again, but this time we added security through WS-Security specification [32], which is responsible for applying cryptography and digital signature to SOAP messages. So there are three different versions of the initial services: one secure, one reliable and a plain version, i.e., without additional parameters. All of these were hosted in and provided by an Apache Tomcat server running on a remote machine.

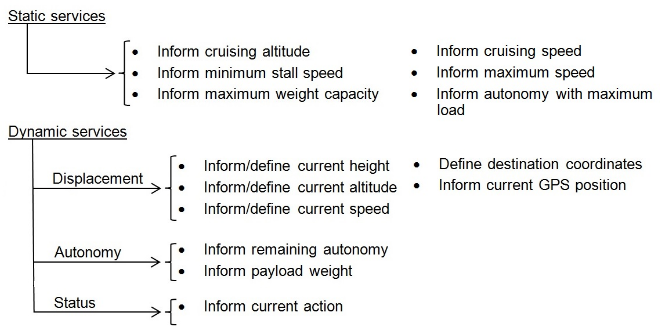
Static services
Inform cruising altitude
Inform minimum stall speed
Inform maximum weight capacity
Inform cruising speed
Inform maximum speed
Inform autonomy with maximum load
Dynamic services
Displacement
Inform/define current height
Inform/define current altitude
Inform/define current speed
Define destination coordinates
Inform current GPS position
Autonomy
Inform remaining autonomy
Inform payload weight
Status
Inform current action

Fig. 5: Basic services implemented in KBF.

On the client side, we implemented an application that makes several calls to those services. The amount of calls varies from one experiment to other. In these experiments, client and server were in different machines at the same local network.

In each experiment, the clients were run several times and the Round Trip Time (RTT) was measured in each repetition. In the end of the experiment, the average RTT was done, as well the standard deviation and confidence interval (95%) in each case for the different versions and then, finally, the results were compared.

In order to compare the performance in terms of RTT of the different versions, we realized an experiment in which a client machine runs a sequence of service requests, and the RTT was measured for different sequence sizes, i.e., for different amounts of services. The results were obtained through 30 repetitions of each experiment.

As shown in Figure 6, for each experiment (different amount of services), the plain version achieved the best performance when compared to the other two versions. Since it has no additional

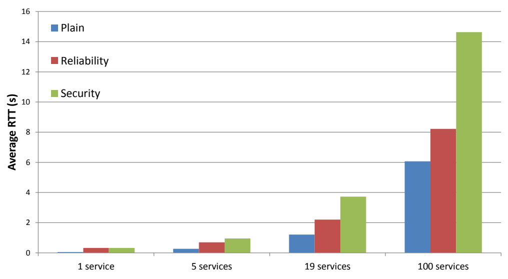

Fig. 6: Average RTT of different amount of services, for different versions.

parameters, it has fewer operations to be executed; therefore it takes less time to be executed.

In the first experiment, with 1 service, both the reliable and the secure versions achieved a similar RTT. Since the secure version uses cryptography and digital signature, it would be usual to think it might have the highest RTT. However, for a single service, the WS-RM exchanges 4 more messages than the other two versions. That is three more times messages in a single services call, and since access to the network is a slow operation, it causes a higher impact on the RTT, making it similar to the impact caused by security operations such as cryptography and digital signature.

In the next experiment, with 5 services, the average RTT increased for all three versions. However, now there is a difference between the secure and reliable version's RTT, because the 4 extra messages of the WS-RM resulted in less than 3 times more messages. Therefore the impact caused by the cryptography and digital signature operations is now greater than the one caused by extra usage of network.

The experiment with 19 services evaluates the behaviour of an application used in a real life situation. As shown in Figure 6, the plain version achieved the best performance, followed by the reliable version in second, and the secure version achieved the highest RTT. Besides, the difference between the reliable and secure versions is greater now than before. Since there are more messages being exchanged, the impact caused by the WS-RM is even smaller.

This pattern can also be observed in the last experiment, with 100 services. The impact of the extra messages sent by the WS-RM specification is even smaller, making the reliable version's RTT more far to the secure version and closer to the plain version.

VI. Conclusions

UAVs are complex systems that perform complex missions. Large UAVs systems are distributed in dozens of different processor systems. The Reference model architecture aims to standardize the various parts that makes up a system. This kind of architecture brings benefits to systems like UAVs, primarily for being safety critical and complex systems. In this sense, LARISSA meets the existing needs.

This paper also introduced the use of SOA in critical embedded systems, providing dynamic behaviour and flexibility to this class of systems. However, the issue of choosing the parts of the system that can be implemented with this technology, without compromising its safety-critical nature, is not a trivial task.

Different types of services, based on avionics, and the effects on performance when using them were also presented. The results showed that by applying security and reliability to these services, a considerable overhead is generated, and so this might cause problems on applications that are exclusively dependent on real time performance. However, considering UAVs, there are occasions in which the real time performance is not the main requirement. Therefore it is possible to use these resources, but it is designer's responsibility to decide where and when to make services secure and/or reliable, and also to decide the level of security required for the mission being developed.

The architecture and the framework presented, backed up by the tests results, allow the use of SOA in the sections of low criticality of safety-critical embedded systems, specially UAVs, leading to a breakthrough in the development of this class of systems, making it easier and more feasible to create, reuse and maintain safety-critical embedded systems.

REFERENCES

[1] L. Lazić and D. Velašević, "Applying simulation and design of experiments to the embedded software testing process: Research articles," *Software Testing, Verification & Reliability*, vol. 14, no. 4, pp. 257–282, Dec. 2004.

[2] A. Armoush, E. Beckschulze, and S. Kowalewski, "Safety assessment of design patterns for safety-critical embedded systems," in *SEAA '09: Proceedings of the 2009 35th Euromicro Conference on Software Engineering and Advanced Applications*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 523–527.

[3] S. P. Kumar, P. S. Ramaiah, and V. Khanaa, "Architectural patterns to design software safety based safety-critical systems," in *ICCCS '11: Proceedings of the 2011 International Conference on Communication, Computing & Security*. New York, NY, USA: ACM, 2011, pp. 620–623.

[4] Z. Yi, W. Cai, and W. Yue, "Adaptive safety critical middleware for distributed and embedded safety critical system," in *NCM '08: Proceedings of the 2008 Fourth International Conference on Networked Computing and Advanced Information Management - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 162–166.

[5] R. Bergamaschi, G. Martin, W. Wolf, R. Ernst, K. Vissers, and J. Kouloheris, "The future of system-level design: Can we find the right solutions to the right problems at the right time?" in *CODES+ISSS '03: Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. New York, NY, USA: ACM, 2003, pp. 231–231.

[6] GAO, "Unmanned aircraft systems: Federal actions needed to ensure safety and expand their potential uses within the national airspace system," 2008, United States Government Accountability Office. Report to Congressional Requesters.

[7] O. Trindade, L. O. Neris, L. C. P. Barbosa, and K. R. L. J. C. Branco, "A layered approach to design autopilots," in *ICIT '10: Proceedings of the IEEE International Conference on Industrial Technology*, march 2010, pp. 1415–1420.

[8] K. R. L. J. C. Branco, J. M. Pelizzoni, L. O. Neris, O. Trindade, F. S. Osório, and D. F. Wolf, "Tiriba - a new approach of uav based on model driven development and multiprocessors," in *ICRA '11: IEEE International Conference on Robotics and Automation*, may 2011, pp. 1–4.

[9] D. Rodrigues, R. M. Pires, J. C. Estrella, M. Vieira, M. Correa, J. B. Camargo, K. R. L. J. C. Branco, and O. Trindade, "Application of SOA in safety-critical embedded systems," *Communications in Computer and Information Science*, vol. 206, pp. 345–354, 2011.

[10] DoD, "Unmanned systems roadmap 2007-2032," 2007, U.S. Department of Defense. Office of the Secretary of Defense.

[11] DoD, "Unmanned systems integrated roadmap FY2009-2034," 2009, U.S. Department of Defense. Office of the Secretary of Defense.

[12] NIST, "4D/RCS: A reference model architecture for unmanned vehicle systems version 2.0," 2002, NIST. U.S. Department of Commerce.

[13] E. Pastor, J. Lopez, and P. Royo, "UAV payload and mission control hardware/software architecture," *IEEE Aerospace and Electronic Systems Magazine*, vol. 22, no. 6, pp. 3–8, 2007.

[14] L. Olson and L. Burns, "A common architecture prototype for army tactical and fcs uavs," DASC '05: The 24th Digital Avionics Systems Conference. Washington,: IEEE, 2005, pp. 8.B.5–1 – 8.B.5–11.

[15] J. Neto, R. Paixao, L. Rodrigues, E. Moreira, J. dos Santos, and P. Rosa, "A surveillance task for a uav in a natural disaster scenario," 2012 IEEE International Symposium on Industrial Electronics (ISIE). Hangzhou: IEEE, 2012, pp. 1516–1522.

[16] P. J. Prisaznuk, "Integrated modular avionics," in *National Aerospace and Electronics Conference (NAECON)*. IEEE, 1992, pp. 39–45.

[17] C. B. Watkins and R. Walter, "Transitioning from federated avionics architectures to integrated modular avionics," DASC '07: 26th Digital Avionics Systems Conference. Dallas, TX, USA: IEEE, 2007, pp. 2.A.1–1 – 2.A.1–10.

[18] W. R. Inc, "Arinc 653: An avionics standard for safe, partitioned systems," in *IEEE-CS Seminar*, 2008.

[19] N. R. Kakanakov, "Experimental analysis of client/server applications in embedded systems," in *Proceedings of the Annual Scientific and Applied Science Conference Electronics*, vol. 4, 2005, pp. 97–102.

[20] N. R. Kakanakov and G. Spasov, "Adaptation of web service architecture in distributed embedded systems," in *CompSysTech '05: Proceedings of the International Conference on Computer Systems and Technologies*, 2005, pp. 1–6.

[21] K. C. Thramboulidis, G. Doukas, and G. Koumoutsos, "A SOA-based embedded systems development environment for industrial automation," *EURASIP Journal on Embedded Systems - Embedded System Design in Intelligent Industrial Automation*, vol. 2008, pp. 1–15, 2008.

[22] M. H. Lee, C. J. Yoo, and O. B. Jang, "Embedded system software testing based on SOA for mobile service," *International Journal of Advanced Science and Technology*, vol. 1, no. 1, pp. 55–64, 2008.

[23] G. Moritz, S. Prüter, D. Timmermann, and F. Golatowski, "Web services on deeply embedded devices with real-time processing," in *ETFA '08: IEEE International Conference on Emerging Technologies and Factory Automation*, 2008, pp. 432–435.

[24] S. Deugd, R. Carroll, K. E. Kelly, B. Millett, and J. Ricker, "SODA: Service oriented device architecture," *IEEE Pervasive Computing*, vol. 5, no. 3, pp. 94–96, 2006.

[25] H. Bohn, A. Bobek, and F. Golatowski, "SIRENA - service infrastructure for real-time embedded networked devices: A service oriented framework for different domains," in *ICNICONSMCL '06: Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 43–48.

[26] A. S. Tanenbaum, *Computer Networks*, 5th ed. Prentice Hall, 1997.

[27] K. P. Valavanis, *Advances in Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy*, ser. International Series on Intelligent Systems, Control, and Automation. Springer, 2007.

[28] R. M. Pires, D. Rodrigues, K. R. L. J. C. Branco, and O. Trindade, "Mosa - mission oriented sensor array: A proposal," in *CLEI '11: Proceedings of the XXXVII Conferencia Latinoamericana de Informática*, 2011, pp. 1309–1318.

[29] D. Rodrigues, R. M. Pires, J. C. Estrella, M. Vieira, M. Correa, J. B. Camargo, K. R. L. J. C. Branco, and O. Trindade, "Application of SOA in safety-critical embedded systems," *Communications in Computer and Information Science*, vol. 206, pp. 345–354, 2011.

[30] D. Rodrigues, R. M. Pires, J. C. Estrella, E. A. Marconato, O. Trindade, and K. R. L. J. C. Branco, "Using SOA in critical-embedded systems," in *Proceedings of the 2011 IEEE International Conferences on Internet of Things (iThings), and Cyber, Physical and Social Computing (CPSCom)*, 2011, pp. 733–738.

[31] OASIS, "Web services reliable messaging (WS-ReliableMessaging) version 1.2," 2009, http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.2-spec-os.html, Accessed 06 April 2014.

[32] OASIS, "Web services security (WSS) TC," 2006, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss, Accessed 06 April 2014.