Independent Configurable Architecture for Reliable Operation of Unmanned Systems with Distributed Onboard Services

Swee Balachandran[1], César A. Muñoz[2], María C. Consiglio[2], Marco A. Feliú[1], and Anand V. Patel[1]

***Abstract*— This paper presents the development of ICAROUS-2 (Independent Configurable Architecture for Reliable Operation of Unmanned Systems with Distributed Onboard Services), the second generation of a software architecture that integrates several algorithms as distributed onboard services to enable robust autonomous UAS applications. In particular, the ICAROUS architecture defines a framework to perform detect and avoid, geofencing, path monitoring, path planning, and autonomous decision making to ensure safety and mission progress. Most of the core algorithms implemented in ICAROUS are formally verified using an interactive theorem prover. These algorithms are composed together using a plan execution engine, whose operational semantics is formally specified. A description of the integrated architecture, services currently available, and flight test results highlighting the capability of ICAROUS are presented.**

I. Introduction

The committee on Autonomy for Civil Aviation published a report [1] on research areas required to facilitate the integration of increasingly autonomous systems in the national airspace. The development of formally verifiable system architectures was identified as a key research area required for the success of these systems. Despite the emphasis on civil aviation, the research areas identified in [1] also apply to small UAS due to the similarity in technological challenges they share. This paper discusses the development of ICAROUS-2 (Independent Configurable Architecture for Reliable Operation of Unmanned Systems with Distributed On-Board Services), a software architecture to build safety-centric autonomous UAS missions.

The growing applications of small UAS invariably result in software that consists of interactions (synchronous and asynchronous) between multiple software applications, sensors and hardware (CPUs and GPUs) components. The increasingly complex use cases for small unmanned aerial vehicles in close proximity to urban environments raise several safety concerns relevant to the airframe hardware, flight control, and decision making software. The definition of models that specify how each software component behaves, interacts with other components and handles exceptions occurring in real time is crucial in establishing the safety of the overall system. Testing each possible execution path of these systems can be challenging and nearly impossible. A complementary technique is the use of formally verified algorithms that, under appropriate operational and environmental assumptions, ensure the correct behavior of the system. Thus, testing can be focused on assuring that the operational and environmental assumptions are realistic.

ICAROUS is a software architecture that integrates several formally verified core algorithms commonly used in UAS operations. The software applications within ICAROUS are functionally distributed into conflict monitors, conflict resolvers, decision makers, and mission-specific functions. Each application within ICAROUS is independent and publishes relevant information to other applications using a software bus. The decision making application processes information from monitoring applications and triggers appropriate resolutions.

Section II provides a review of relevant architectures found in the literature. Section III describes the functional layout of the ICAROUS architecture and the middleware currently being used. Section IV describes the various onboard services that ICAROUS integrates to enable autonomous UAS missions. Section V provides flight test results demonstrating the capabilities of ICAROUS. Finally, section VI provides a discussion on the ICAROUS architecture with potential future work directions. Finally section VII provides conclusions.

II. Related Work

Several government, industry, and research organizations have recognized the importance of having an architecture standard to promote the development, reuse, and interoperability of software/hardware components across a wide range of unmanned systems operating and collaborating over land, air and water. The US Department of Defense (DoD) calls for the adoption of a modular Open Systems Architecture (OSA) and consequently several organizations have developed various architectures consistent with the OSA concept. A few examples are the Unmanned Systems Command and Control Standard Initiative (UCI), Open Mission Systems (OMS), and the Future Airborne Capability Environment (FACE). A detailed discussion and comparison between these various OSA efforts is described in [2]. Reference [3] provides a comprehensive survey of various architectural developments spearheaded by DoD, the US Navy, and the US Air Force.

Pastor et al. [4] proposed the use of a Service Oriented Architecture (SOA) with a UAV service abstraction layer that enables high-level services to provide flight control inputs. The various services were categorized into one of the following categories: mission, payload, flight, awareness. How et al. [5] developed and implemented the MIT CSAT (Cooperative Search, Acquisition and Track) Architecture. Bamberger et al. [6] developed the John Hopkins APL

[1]National Institute of Aerospace, Hampton, Virginia 23666. swee.balachandran@nianet.org.

[2]NASA Langley Research Center, Hampton, Virginia 23681. cesar.a.munoz@nasa.gov.

Autonomous UAV Architecture with a focus of controlling a swarm of UAVs. Curtis Heisey et al. [7] developed the MIT/LL Reference Architecture for small UAS.

The Air Force Research Laboratory developed the OpenUxAS system [8]. OpenUxAS is a collection of modular services to perform surveillance missions with one or more UASs. A process algebra framework enables the specification of tasks. These tasks are then distributed optimally among multiple UAS which then collaborate to efficiently complete the task.

III. ICAROUS Architecture

A. Functional architecture

The ICAROUS architecture is similar in spirit to the SOA architecture proposed by Pastor et al. [4]. A set of services provide various capabilities such as path planning, sense and avoid, geofence containment, task planning, etc. These services are commonly used to construct complex autonomous UAS applications. Unlike the architectures discussed in the preceding section, the ICAROUS architecture also specifies a framework for performing conflict monitoring and resolutions. Furthermore, a decision making application coordinates the output of conflict monitors and resolvers to ensure safe operation.

Shown in Figure 1 is a description of the functional layout of the ICAROUS architecture. Applications are logically organized into conflict monitors, conflict resolvers, mission managers, and decision makers.

Conflict monitors are algorithms that monitor for imminent violation of airspace constraints such as geofences, conflicts due to other vehicles in the airspace, deviations from mission flight plan, etc. These conflict monitoring applications can also provide tactical resolutions. A tactical resolution is a simple maneuver that, if executed, is guaranteed to prevent the corresponding conflict violation. However, a tactical resolution does not predict future conflict violations and may not resolve other types of conflicts if they exist simultaneously. The notion of a conflict is abstractly represented by a set of descriptors that convey meta level information about a conflict such as its severity, time to conflict violation, and point of no return. These descriptors enable a decision making tool to sort them in an appropriate order to suite the mission needs and consequently invoke the corresponding resolvers.

Conflict resolvers compute resolutions to prevent imminent violation of specified constraints. There can be several resolvers, one for each conflict detector. Resolvers may also handle multiple conflicts simultaneously. Resolvers provide strategic resolutions that are computed to prevent one or more constraint violations. A resolution is also abstractly represented by descriptors that provide meta level information about the resolution such as resolution type, time to recovery, etc.

A decision making application receives conflict information from monitors and triggers resolvers to compute resolutions for one or more conflicts. When resolving imminent constraint violation, outputs from mission applications

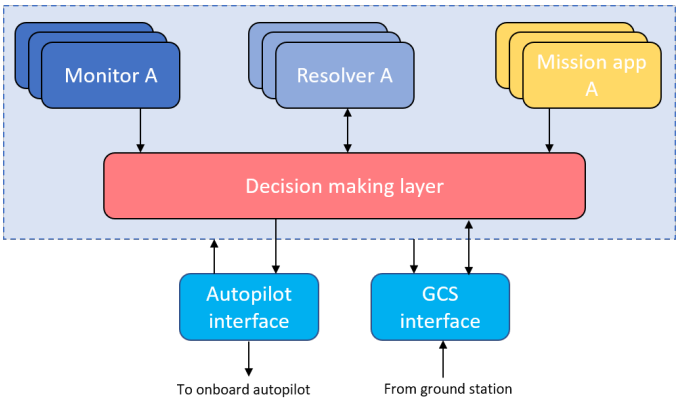

Fig. 1. Functional architecture

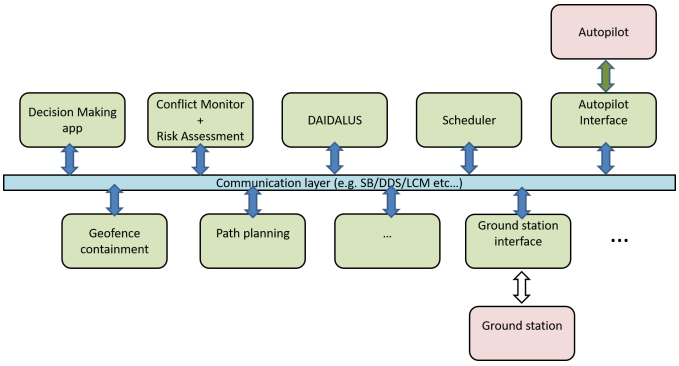

Fig. 2. ICAROUS from a Service Oriented Architecture perspective (Figure is notional)

are ignored. The mission is resumed once all conflicts are resolved.

B. Middleware architecture

The presence of various hardware devices for embedded systems, associated variability in hardware architectures, and operating systems requires the use of a middleware to abstract away from the underlying operating system and hardware architecture. Furthermore, the complexity of UAS missions which require interactions between various software applications require the existence of a robust communication framework. ICAROUS is currently being developed using the NASA core Flight Systems (cFS) middleware.

The core Flight System (cFS) is a platform independent reusable software framework and a set of reusable software applications. There are three key aspects to the cFS architecture: a dynamic run-time environment, layered software, and a component-based design. These key aspects make cFS suitable for reuse on any number embedded software systems. The cFS flight software framework takes advantage of a rich heritage of successful Goddard Space Flight Center flight software efforts and addresses the challenges of rapidly increasing software development costs and schedules due to the constant changes and advancements in hardware. Flight software size and complexity is expected to grow dramatically in coming years and the cFS provides a means

to manage the growth and partition complexity. To support reuse and project independence, the architecture contains a configurable set of requirements and code. The configurable parameters allow the cFS to be tailored for each environment including desktop and closed loop simulation environments. The ability to run and test software applications on a developer's desktop and then deploy that same software without changes to the embedded system is possible using the cFS. Science and mission software can be developed and functionally tested very early in the project and well before any project hardware is even available. The cFS provides a tool suite which includes a reusable test suite. In addition, cFS also contains reusable artifacts including requirements, design documentation, test procedures, development standards, and user guides. The cFS middleware simplifies the flight software development process by providing the underlying infrastructure and hosting a runtime environment for development of project/mission specific applications. The cFS architecture also simplifies the flight software maintenance process by providing the ability to change software components during development or in flight without having to restart or reboot the system.

IV. Core Service Description

Each core functionality within ICAROUS can be viewed as a service and is implemented as a cFS application. Each cFS application makes use of the core flight executive features. For example, the applications can exchange information between each other using a publish/subscribe capability provided by the cFS Software Bus. The core ICAROUS services are described in the following sections.

A. Geospatial conformance

A geofence application provides functionalities to monitor imminent conflicts related to keep-in and keep-out geofences. Underlying the geofence application is a library of formally verified core algorithms called PolyCARP [9], [10]. PolyCARP is used to check if a given point is within or outside a polygon. Using a raycasting algorithm as shown in Figure 3, a given point is determined if it is located within/outside a polygon. Information regarding imminent violation with respect to available fences, time to violation and a safe recovery position are published by the geofence monitoring application.

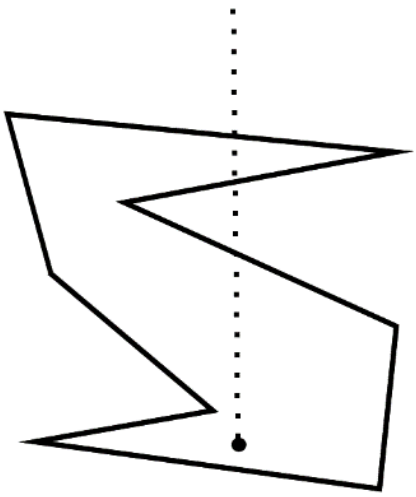

Fig. 3. Ray casting

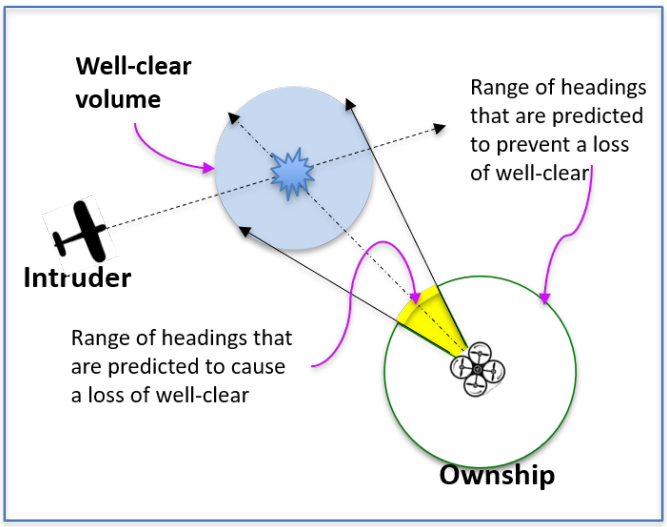

Fig. 4. Detecting and avoiding well clear violations

B. Detect and avoid

A traffic application provides basic avoidance capabilities to check for imminent well clear violation against other vehicles in the airspace. Underlying this application is a suite of formally verified core algorithms referred to as DAIDALUS [11]. DAIDALUS also provides various resolutions (track, ground speed, altitude and vertical speed) that if executed by the vehicle ensures separation among vehicles in the airspace.

C. Path planning

A trajectory application provides path planning capabilities. Various path planning tools based on search and numerical optimization techniques such as A*, RRT, and B-splines are available [12]. Each trajectory planner has varying capabilities. These planners can take into account static obstacles such as geofences and dynamic obstacles such as traffic. Figure 5 compares the outputs of various planners currently available. The trajectory application also monitors for flight plan deviations from the original mission flight plan.

D. Decision making

Decision making in ICAROUS is currently governed by finite state machines implemented using the NASA Plan Execution Interchange Language (PLEXIL) [13]. The operational semantics of PLEXIL has been formally specified and serves as reference implementation of PLEXIL's executive [14]. Decision making is decomposed based on the

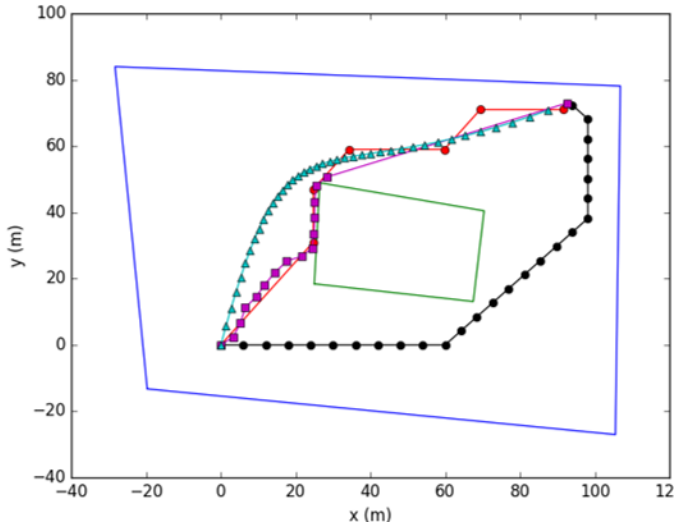

Fig. 5. Comparison of planner outputs for simple geofence re-route scenario (red: Grid A*,black: Trim A*, magenta: RRT, cyan: B-splines)

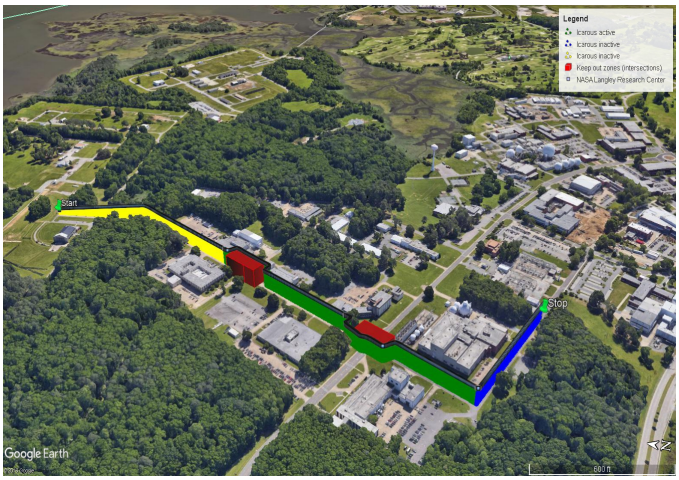

Fig. 6. Flight path for a beyond visual line of sight mission. Takeoff position (top left) and landing position (bottom right). No fly zones around intersections.

various flight phases, i.e., takeoff, climb, cruise, approach, and landing. This leads to a hierarchical set of finite state machines used for decision making. These finite state machines are driven by the outputs of various conflict monitors. Consequently, the finite state machines trigger the planning application to compute alternate paths/reroutes to prevent constraint violations.

E. Mission-Specific functions

In addition to the above core services available, The ICAROUS software suite provides application for mission-specific behaviors such as target tracking, inspecting objects, etc. The decision making application overrides the mission specific behavior if conflict violations are detected. Mission functionality is resumed once all conflicts are resolved.

V. Flight Test Results

The ICAROUS-2 architecture is currently being developed under the auspices of the NASA Unmanned Air Traffic Management (UTM) project. A set of milestones called Technology Capability Levels (TCL) has been established to demonstrate the technologies under development. Several flight tests illustrating the capabilities of ICAROUS have been conducted. Detailed descriptions of these tests can be found in [15]. Significant tests showcasing ICAROUS geofencing and traffic avoidance capabilities are briefly highlighted in this paper.

A. Illustration of geofencing capabilities

Figure 6 illustrates a beyond visual line of sight mission flying through an urban environment and across multiple intersections in the NASA Langley Reesearch Center. The intersections were treated as no fly zones. The flight waypoints were intentionally chosen to fly through the no fly zones. The intersections were represented as keep-out geofences (shown as red boxes in Figure 6). The goal of this test was to let ICAROUS recognize the presence of a keep-out geofence conflict and take necessary actions to prevent constraint violation. Figure 6 illustrates the final flight path of the vehicle. At both intersections, ICAROUS detects the conflict and reroutes the vehicles around the intersection. The green segments in Figure 6 indicate the portions of the flight when ICAROUS was active.

B. Illustration of traffic avoidance capabilities

ICAROUS traffic avoiding capability was demonstrated against stationary and moving intruders. A cylindrical well clear volume of radius 12 m and height 100 m was chosen. For safety reasons, intruder vehicles were flown with an altitude separation. Vehicles communicated their position to each other using Dedicated Short Range Communication (DSRC) radios. Figure 7 illustrates a scenario where ICAROUS encounters an intruder that remained stationary on the ground. Figure 7 also illustrates the track resolutions executed by ICAROUS at two different points during the encounter. The horizontal distance to the intruder as a function of time is illustrated in figure 8. Figure 9 illustrates the conflicting track angles that could result in loss of separation with the intruder vehicle. ICAROUS maneuvers the vehicle to stay clear of these track angles.

VI. Discussion

The architecture discussion in the preceding sections depends on monitoring services to detect conflicts. Consequently, this requires appropriate conflict detectors to monitor specified constraints. Currently, the ICAROUS architecture supports detecting conflicts due to geofence (keep-in/keep-out) constraints, well-clear violations due to other intruders in the airspace, and flight plan deviation. To address any other conflicts, appropriate monitors must be incorporated. For example, unsafe conditions due to off-nominal battery voltages can be flagged by a better health monitoring service. The abstract representation of conflicts defined in the ICAROUS architecture enables expanding the suite of

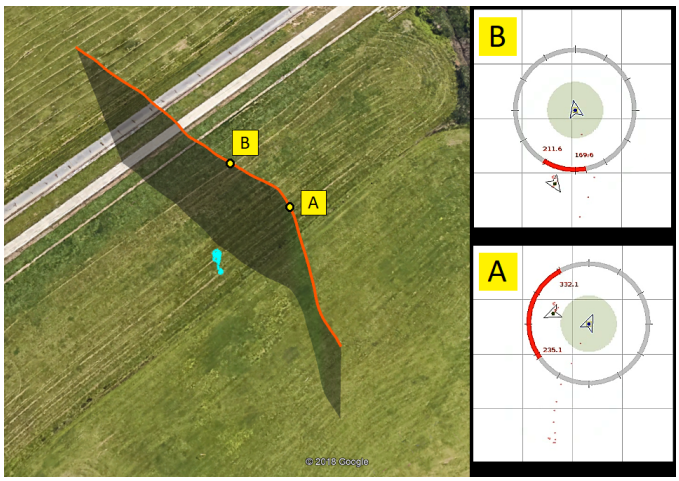

Fig. 7. Encounter with a stationary intruder

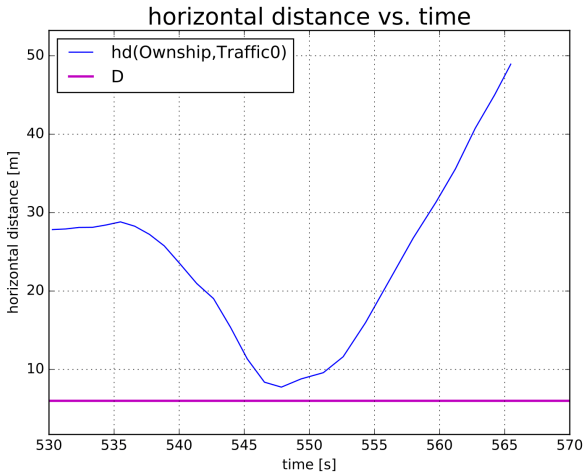

Fig. 8. Horizontal distance between ownship and stationary intruder

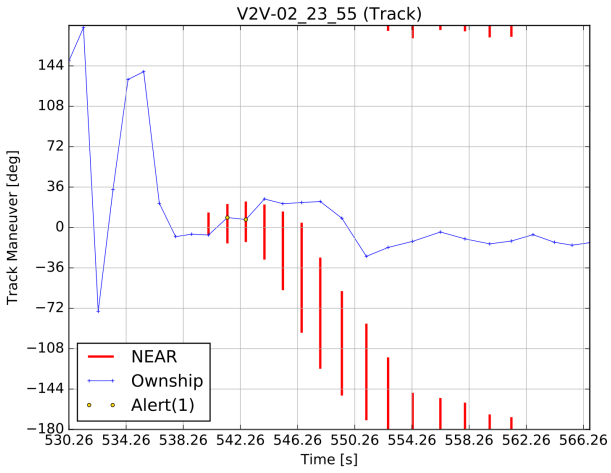

Fig. 9. Track angles that could result in loss of separation

monitoring applications. Adding new monitors also requires having appropriate conflict resolvers to handle new conflicts.

VII. Conclusion

This paper presented the ICAROUS-2 architecture. ICAROUS-2 follows the service oriented architecture paradigm and provides a logical organization of applications into conflict monitors, resolvers, mission managers, and decision making tools to build safe UAS missions. The current implementation of ICAROUS-2 provides several monitoring and resolving services for geofence conformance and traffic avoidance. Several of the core algorithms used by the conflict detectors and resolvers are formally verified using interactive theorem provers. Future work will look into expanding the core functionality of ICAROUS with more features, conflict monitors (e.g., battery health monitor) and safe flight-termination services.

References

[1] N. R. Council *et al.*, *Autonomy research for civil aviation: toward a new era of flight*. National Academies Press, 2014.

[2] J. L. Tokar, "A comparison of avionics open system architectures," *ACM SIGAda Ada Letters*, vol. 36, no. 2, pp. 22–26, 2017.

[3] D. Gonzales and S. Harting, "Designing unmanned systems with greater autonomy: using a federated, partially open systems architecture approach," RAND NATIONAL DEFENSE RESEARCH INST SANTA MONICA CA, Tech. Rep., 2014.

[4] E. Pastor, C. Barrado, E. Santamaria, J. Lopez, and P. Royo, *An open architecture for the integration of UAV civil applications*. Citeseer, 2009.

[5] J. P. How, C. Fraser, K. C. Kulling, and L. F. Bertuccelli, "Increasing autonomy of uavs," *IEEE Robotics & Automation Magazine*, vol. 16, no. 2, 2009.

[6] R. J. Bamberger Jr, D. P. Watson, D. H. Scheidt, and K. L. Moore, "Flight demonstrations of unmanned aerial vehicle swarming concepts," *Johns Hopkins APL technical digest*, vol. 27, no. 1, pp. 41–55, 2006.

[7] C. W. Heisey, A. G. Hendrickson, B. J. Chludzinski, R. E. Cole, M. Ford, L. Herbek, M. Ljungberg, Z. Magdum, D. Marquis, A. Mezhirov *et al.*, "A reference software architecture to support unmanned aircraft integration in the national airspace system," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 41–55, 2013.

[8] L. H. Steven Rasmussen, Derek Kingston, "A brief introduction to unmanned systems autonomy services (uxas)," in *2010 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2018.

[9] A. Narkawicz and G. Hagen, "Algorithms for collision detection between a point and a moving polygon, with applications to aircraft weather avoidance," in *16th AIAA Aviation Technology, Integration, and Operations Conference, AIAA AVIATION Forum*, no. AIAA-2016-3598, Washington, DC, USA, June 2016. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2016-3598

[10] A. Narkawicz, C. Muñoz, and A. Dutle, "The MINERVA software development process," in *Automated Formal Methods*, ser. Kalpa Publications in Computing, N. Shankar and B. Dutertre, Eds., vol. 5. EasyChair, 2018, pp. 93–108. [Online]. Available: https://easychair.org/publications/paper/g1Rs

[11] C. Muñoz, A. Narkawicz, G. Hagen, J. Upchurch, A. Dutle, and M. Consiglio, "DAIDALUS: Detect and Avoid Alerting Logic for Unmanned Systems," in *Proceedings of the 34th Digital Avionics Systems Conference (DASC 2015)*, Prague, Czech Republic, September 2015.

[12] S. Balachandran, A. Narkawicz, C. Muñoz, and M. Consiglio, "A path planning algorithm to enable well-clear low altitude UAS operation beyond visual line of sight," in *Proceedings of the 12th USA/Europe Air Traffic Management R&D Seminar, ATM 2017*, no. 16, Seattle, Washington, 2017.

[13] V. Verma, T. Estlin, A. Jónsson, C. Pasareanu, R. Simmons, and K. Tso, "Plan execution interchange language (plexil) for executable plans and command sequences," in *International symposium on artificial intelligence, robotics and automation in space (iSAIRAS)*, 2005.

[14] G. Dowek, C. Muñoz, and C. Rocha, "Rewriting logic semantics of a plan execution language," *Electronic Proceedings in Theoretical Computer Science*, vol. 18, pp. 77–91, 2010.

[15] A. Moore, S. Balachandran, S. D. Young, E. T. Dill, M. J. Logan, L. J. Glaab, C. Munoz, and M. Consiglio, "Testing enabling technologies for safe uas urban operations," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3200.