A Graphical Toolkit for the Validation of Requirements for Detect and Avoid Systems*

Paolo Masci[1] and César A. Muñoz[2]

[1] National Institute of Aerospace, Hampton, VA, USA
paolo.masci@nianet.org
[2] NASA Langley Research Center, Hampton, VA, USA
cesar.a.munoz@nasa.gov

Abstract. Detect and Avoid (DAA) systems are safety enhancement software applications that provide situational awareness and maneuvering guidance to aid aircraft pilots in avoiding and remaining well clear from other aircraft in the airspace. This paper presents a graphical toolkit, called DAA-Displays, designed to facilitate the assessment of compliance of DAA software implementations to formally specified functional and operational requirements. The toolkit integrates simulation and prototyping technologies allowing designers, domain experts, and pilots to compare the behavior of a DAA implementation against its formal specification. The toolkit has been used to validate an actual software implementation of DAA for unmanned aircraft systems against a standard reference algorithm that has been formally verified.

Keywords: Validation · Verification · Requirements · Detect and Avoid · Formal Methods

1 Introduction

Aircraft pilots operating under visual flight rules, including pilots of remotely operated vehicles, have the legal responsibility to see and avoid other aircraft in the airspace [17, 18]. In the case of manned aircraft operations, the ability to remain well-clear and see and avoid other aircraft depends upon the perception and judgement of the human pilot. In the absence of an on-board pilot, there is a need for an objective definition of the notion of well-clear that is appropriate for Unmanned Aircraft Systems (UAS). This need has motivated the development of a Detect and Avoid (DAA) capability for UAS that provides situational awareness and maneuver guidance to UAS operators, to aid them in avoiding and remaining well-clear of other aircraft in the airspace [3].

The RTCA[3] standard document DO-365 [15] specifies the minimum operational and functional DAA requirements for large UAS, e.g., those that fly in

* Research by first author was supported by the National Aeronautics and Space Administration under NASA/NIA Cooperative Agreement NNL09AA00A.

[3] RTCA was formerly known as Radio Technical Commission for Aeronautics.

Class A airspace. DAIDALUS (Detect and Avoid Alerting Logic for Unmanned Systems) [11, 13] is an open source software library[4] that implements the functional requirements specified in DO-365. The core algorithms in DAIDALUS, including detection, alerting, and maneuver guidance logics, are formally verified in the Prototype Verification System (PVS) [14]. Similar standardization efforts are currently under way for small UAS. In the case of general aviation, the DAA in the Cockpit (DANTi) concept [1, 2] developed at NASA leverages advancements in DAA technologies for UAS as a safety enhancing capability for pilots flying under visual flight rules and who are not receiving Air Traffic Control radar services. The DAIDALUS library can be configured to support DAA capabilities for all those operational cases, i.e., small to large UAS and general aviation aircraft.

DAA systems use aircraft state information, e.g., position and velocity 3-D vectors, to predict a loss of well-clear between the primary vehicle, known as the *ownship*, and traffic aircraft, known as *intruders*. In case of a predicted loss of well-clear between the ownship and an intruder aircraft, an alert level is generated. The alert level is an indication of the severity of the predicted loss assuming the ownship and the intruder do not maneuver. Depending on the alert level maneuver guidance is provided to the ownship to avoid the intruders. Maneuver guidance has the form of *bands*, i.e., ranges of heading, horizontal speed, vertical speed, and altitude maneuvers that are predicted to be conflict free. The determination of the well-clear status is based on a mathematical formula that uses distance and time separation thresholds. The actual values of these thresholds depend on the operational case. For instance, in DO-365, the minimum separation is 4000ft horizontally and 450ft vertically. Furthermore, there is a time component that accounts for encounters with a high closure rate. That time component, which is an approximation of the time to closest point of approach, is 35s. These thresholds define a volume in space and time, called *well-clear volume*, that has the shape of a cylinder elongated in the direction of the relative velocity between the ownship and the intruder [10]. The thresholds for small UAS are smaller and the definition does not include a time component. For general aviation, the thresholds are slightly larger than for UAS.

In addition to minimum operational and functional requirements, DO-365 provides a set of test vectors intended to facilitate the validation of DAA implementations against these requirements. DAA developers may use DAIDALUS, the reference implementation, to *validate* their DAA implementations against the test vectors. If the two systems provide the same outputs for all test vectors, then confidence is gained in the functional correctness of the implementation. While this validation approach based on systematic comparison with a reference specification is conceptually simple, it poses some key challenges. One main hurdle originates from round-off errors in machine arithmetic. In fact, DAA implementations, including those developed for DAIDALUS, use floating-point arithmetic. The verified algorithms used in DAIDALUS, on the other hand, are specified in PVS, and use real arithmetic. When numeric computations are used in control

[4] https://shemesh.larc.nasa.gov/fm/DAIDALUS.

flows, round-off errors introduced by floating-point arithmetic may alter the logic of a real-valued algorithm.

A precise characterization of all possible differences produced by round-off errors is not trivial (see, for example, [16]). In the case of DAA functions, round-off errors can introduce delays in the time when a maneuver guidance is provided, or even change the alert level. The net result is that validation approaches based on simple comparison of numerical values are often *inconclusive*, in the sense that numerical differences between the output of a DAA implementation and that of the reference specification do not necessarily flag problems in the implementation. It is also true that, depending on the considered scenario, small numerical differences may flag actual implementation problems.

To assess compliance with the reference specification, a more empirical method can be adopted in addition to numerical comparisons. The method addresses the following question: *"If domain experts look at the maneuver guidances and alert levels provided by both the DAA implementation and the reference specification, would they judge the information provided by the two systems to be the same?"* The work presented in this paper introduces a toolkit, DAA-Displays, that can be used to answer such an empirical question.

Contribution. This paper introduces a toolkit, DAA-Displays, for the validation of DAA implementations. The toolkit can be used by software developers to validate a DAA implementation against a reference specification. It can also be used by domain experts that design and develop DAA requirements for operational concepts, to validate DAA requirements. The toolkit is freely available on GitHub[5] under the NASA Open Source License Agreement.

2 DAA-Displays

DAA-Displays is a graphical toolkit for the design and analysis of DAA implementations and requirements. The toolkit provides three main functionalities:

1. **Rapid prototyping** of cockpit displays with DAA functions;
2. **Split-view simulations** for the validation of DAA implementations against reference specifications;
3. **3D simulations** of flight scenarios with aircraft using DAA functions.

2.1 Rapid Prototyping

The rapid prototyping functionalities allow formal methods experts to create realistic simulations suitable to discuss DAA algorithms with a multi-disciplinary team of developers. This feature is particularly useful when executable formal specifications of DAA functional requirements are available. In this case, DAA-Displays enables the visualization of the behavior of the formal specifications on concrete flight scenarios. This way, team members who may not be familiar

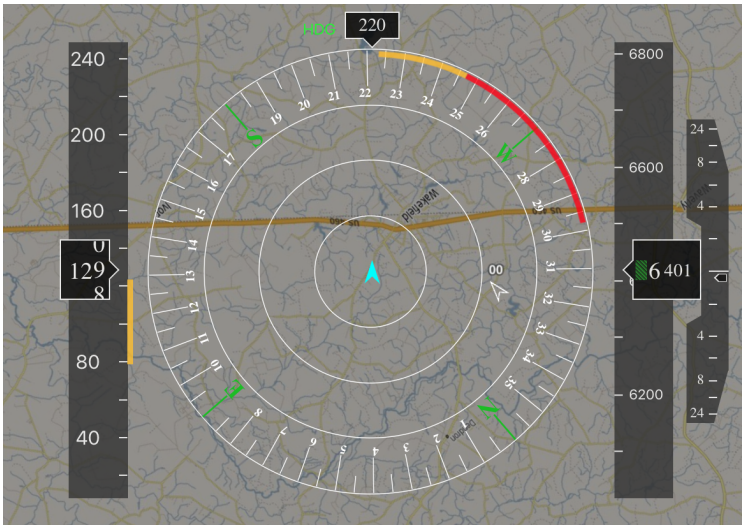

Fig. 1: DAA Cockpit Display

with formal methods can get a better understanding of the type of alerting and maneuver guidance provided by a DAA logic.

An example prototype used for this purpose is in Figure 1. This prototype was created with the DAA-Displays toolkit to discuss different DAA configurations with domain experts and pilots. The prototype reproduces the look and feel of a real cockpit display. It includes the following elements:

- An *interactive map* showing the position and heading of the ownship in the airspace (the blue chevron at the center of the map) as well as the position and heading of traffic aircraft (the other chevrons in the map) relative to the ownship. Color-coded chevrons are used to denote alert levels. These colors are specified in standard documents. For example, in DO-365, a yellow chevron denotes a *corrective* alert level between the ownship and the aircraft. The color red denotes a *warning* alert level. Labels next to the chevrons show the relative altitude of the aircraft with respect to the ownship in hundreds of feet, e.g., 00 indicates co-altitude.
- A *compass* over the map indicates the heading of the ownship. Heading maneuver guidance is displayed on the compass. For example, the yellow and red bands shown on the compass in Figure 1 indicate that the current heading, 220 degrees, is conflict free, but a small change to the right will potentially create a conflict with the traffic aircraft. Similar to alert levels, these bands are color-coded, where yellow denotes a corrective maneuver and red denotes a warning maneuver. According to DAA requirements, bands and alert colors should correspond in the sense that if a traffic aircraft is represented by a

[5] https://github.com/nasa/daa-displays

chevron of a certain color, there should be a band, in the current trajectory of the aircraft, of the same or higher level, e.g., warning is higher than corrective. In Figure 1, if the ownship maneuvers to the right, e.g., 230 degrees, the traffic aircraft becomes yellow. If the ownship maneuvers to 260 degrees, the traffic aircraft becomes red.
- *Tape indicators* at the two sides of the display provide information for airspeed (indicator on the left) in knots, altitude in feet (large indicator on the right), and vertical speed (small indicator on the right) in feet per minute of the ownship. Maneuver guidance involving change of airspeed, altitude, and vertical speed are represented by colored bands over these indicators. For example, the yellow band shown in Figure 1 for the airspeed indicator states that airspeeds in the range 80 to 120 knots would create a potential conflict with the traffic aircraft.

The display elements described above are available in the toolkit in the form of a library of widgets. Additional details on the full set of widgets available in the library are provided in the tool documentation.[6]

2.2 Split-View Simulations

Split-view simulations can be used to visually and systematically compare the output of two DAA logics on the same encounter. The outputs can be from different implementations and executable formal models, or from the same implementation/formal model but with different configuration parameters.

An example split-view simulation used for this purpose is shown in Figure 2. The view includes the following elements:

- *Cockpit displays* show alert levels and maneuver guidance by two DAA implementations for a given encounter and selected configurations at a given moment in time.
- *Spectrogram plots* show alert levels and maneuver guidance as they vary with time; the x-axis in each plot represents time and the y-axis indicates alert levels and maneuver guidance ranges.

The cockpit displays include information about the flight scenarios but focus on single time instants. Spectrogram plots, on the other hand, focus on the temporal behavior of the DAA logics, and give insights on the evolution of alert levels and maneuver guidance over time for a given encounter. For example, with reference to Figure 2, visual inspection of the plot diagrams allows one to confirm that the two DAA implementations under analysis generate maneuver guidance that, judged by a domain expert, are "sufficiently similar," even though the numerical values produced by the two implementations are slightly different in all time instants (as highlighted by the yellow markers at the bottom of the plots).

By inspecting the plot diagrams in Figure 2, one can also notice a delay in changing a red alert to a yellow alert at about 75 seconds. The relevance of

[6] http://shemesh.larc.nasa.gov/fm/DAA-Displays.

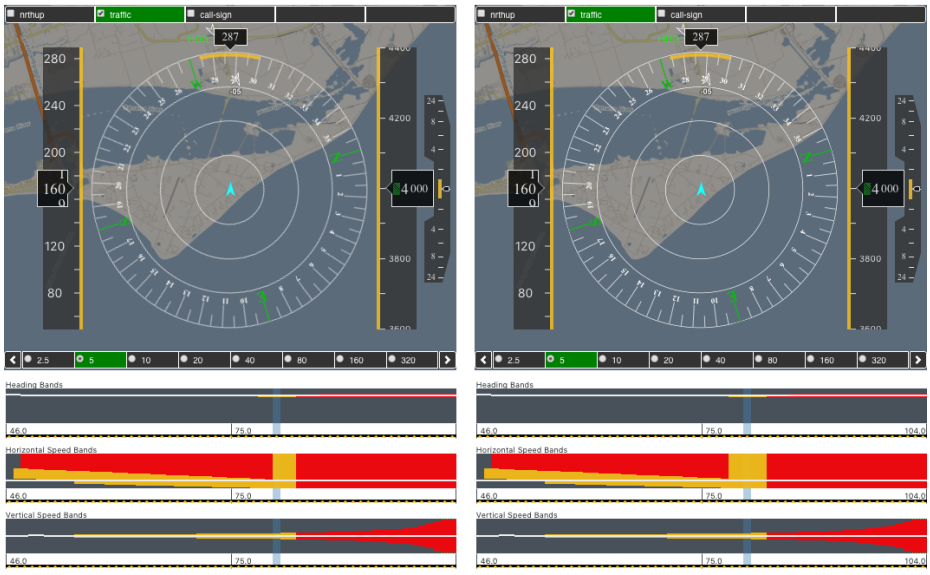

Fig. 2: Example split-view simulation created with DAA-Displays.

Fig. 3: Example 3D simulation created with DAA-Displays.

these kinds of differences may need careful assessment, and typically involves engaging with pilots or DAA designers. This assessment can be carried out with this same split view simulation, as the cockpit displays embedded in the view show the flight scenario in a form that can be understood by both pilots and DAA designers. The simulation is interactive, therefore one can easily jump to specific time instants, and play back fragments of the scenario that are deemed important for the assessment.

2.3 3D Simulations

The 3D simulation capability of the toolkit moves the focus of the analysis from a cockpit-centric view to a scenario-centric view that includes the wider airspace around the ownship. The viewport can be adjusted by tilting, panning, and zooming the view. This capability can be used by developers to gain a better understanding of spatial information on the trajectories followed by the ownship

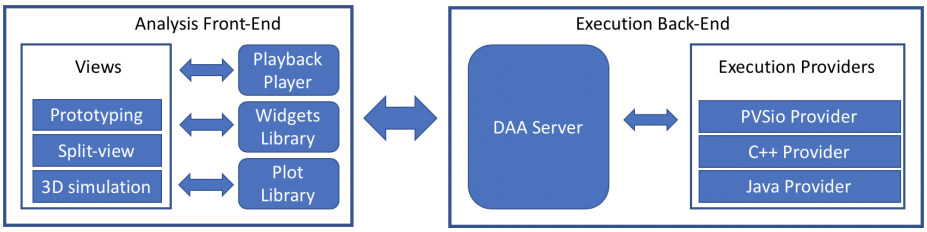

Fig. 4: Architecture of the DAA-Displays toolkit.

and traffic aircraft in a given scenario. This is useful, e.g., when assessing DAA algorithms with computer-generated flight scenarios, as this view provides a tangible idea of what the scenario is about.

An example 3D simulation realized to examine a computer-generated flight scenario is showed in Figure 3. It includes two drones flying over a terrain. The ownship is flying at an altitude of 30 feet. The other drone is following the ownship. The two drones are at the same altitude.

3 Architecture

The architecture of the toolkit is shown in Figure 4. Three main *views* are used to present the functionalities of the toolkit to developers. Underneath, a number of components are used to implement the functionalities of the views. These components can be customized and reused to create new views. A client-server architecture is used to create a separation of concerns between visual components necessary for interactive analysis of DAA functions, and functional components necessary for the execution of DAA specifications and implementations.

Analysis front-end. The analysis front-end constitutes the client side of the architecture. It builds on Web technologies, as this makes it easier to deploy on different platforms, including tablets and mobile devices. The front-end element is implemented in TypeScript, a strict superset of JavaScript annotated with type information. This element includes three main reusable components:

- A *playback player* providing interactive controls for navigating simulation scenarios (e.g., jump to specific time instants and playback of scenarios);
- A *widgets library* containing a series of interactive display elements that can be used to assemble realistic cockpit display simulations;
- A *plot library* providing functionalities for creating interactive plots suitable for rendering alerts and maneuver guidance computed by a DAA implementation over time.

Execution back-end. The execution back-end includes two main components:

- A *DAA Server* implementing communication mechanisms necessary to exchange simulation events and data with the analysis front-end;

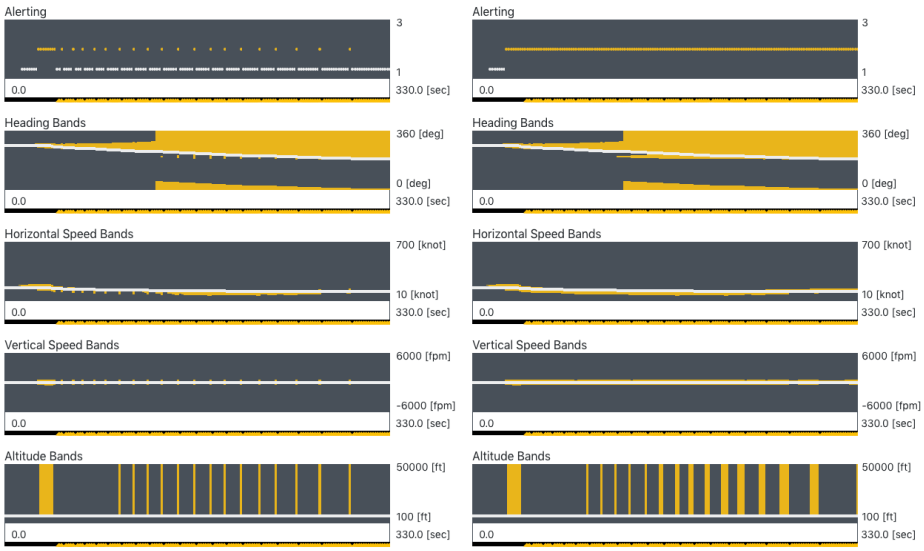

Fig. 5: Spectrogram plots for the analysis of alert flickering.

– An array of *Execution Providers* designed to connect the DAA Server to the native execution environments necessary for the evaluation of DAA specifications and implementations. Each provider implements a standard interface that enables the communication between a given execution environment and the DAA Server. It also incorporates functions for automatic testing of properties that should always be true during the execution of DAA specifications and implementations.

4 Use Cases

The toolkit is currently used to support the development of NASA's DANTi concept [1, 2] and the development of operational and functional requirements for large UAS in landing and departing operations. These latter requirements are being defined by RTCA Special Committee 228 and will be included in the upcoming revision of DO-365. Example analyses and findings are discussed in the remainder of this section.

Alert Flickering. A number of parameters can be used to configure when and how alerting and maneuver guidance are computed in DAIDALUS. The toolkit was used to gather additional insights on corner cases identified for certain configurations. An example corner case relates to *alert flickering*, i.e., situations where an alert level intermittently changes from one second to the next. This unintended behavior has been detected in certain scenarios for specific configurations.

Split-view simulations proved useful for the identification of these problems and for the development of possible solutions. In particular, spectrogram plots

helped with the identification of flickering in alert levels and spikes in maneuver guidance. An example split-view simulation carried out to evaluate a DAA algorithm providing a possible solution to alert flickering is given in Figure 5. The simulation shows alerts and bands computed by the original algorithm (diagrams on the left-hand side of the figure) against those computed by a new algorithm that uses hysteresis (diagrams on the right side). As it can be seen in the *Alerting* plot (first plot from the top), alerts are not toggling in the new algorithm. This solution, however, does not mitigate all problems. In fact, altitude bands are still toggling — this can be easily seen from the spikes in the altitude bands plot at the bottom of Figure 5.

Quick identification of these kinds of shortcomings at the early stage of design of new algorithms is key to speed up development. The sheer use of simulation and visualization technologies was sufficient to identify this issue on the spot. It can be argued that this behavior would have been eventually discovered with formal proofs. However, in this particular case, flickering is caused by small numerical errors in the computation of alerts and maneuver guidance. These kinds of errors are difficult to find and fix in a formal setting, as they often require floating point round-off error analysis.

DANTi Display. The prototyping capabilities of the toolkit are currently being used to support the development of a cockpit display for the DANTi concept. Different prototypes have been developed in rapid succession to explore display layouts and functionalities.

The initial DANTi prototype is shown in Figure 1. The new version introduces new visual elements on the display for changing the zoom level of the map (buttons at the bottom of the display) and for selecting the level of details shown for traffic information (buttons at the top of the display). Future versions, currently under development, include rendering of virtual regions in the airspace (geo-fences) as well as aircraft trajectories. All these prototypes can be deployed on portable electronic flight bags that can be carried by pilots and can be employed for user evaluations, e.g., to perform acceptability studies where pilots are asked to assess the utility of the display in enhancing their see-and-avoid capabilities.

Since the prototypes created with DAA-Displays can be driven by formal specifications this opens the possibility to the use of formal models directly in user evaluations, removing the burden of creating implementations that mimic the formal specifications.

5 Related Work

PVSio-web [9] is a formal methods framework for modeling and analysis of interactive systems. That framework has been extensively used for the analysis of medical devices [8]. The toolkit presented in this work builds on experiences with developing and using that framework. The main design aspect inherited from PVSio-web is the architecture for linking interactive prototypes to

executable formal specifications. New design concepts are introduced in DAA-Displays that are not present in PVSio-web such as split-view simulation components, a widgets library of cockpit display elements, and 3D visualization capabilities. SCADE Displays [7] is a prototyping tool for cockpit displays. The tool can be used to create cockpit display prototypes suitable to visualize maneuver recommendations. However, mechanisms are not provided for linking the prototype to formal specifications. SCR [6] is a toolset for the analysis of software requirements. Prototyping functionalities are supported, and the prototypes can be linked to executable formal models. However, SCR does not support split-view simulations facilitating systematic comparison of different implementations. PVSioChecker [5] and MINERVA [12] are formal methods tools for comparative analysis of PVS specifications and software code. These tools, however, focus on checking *numerical differences* between the output produced by the implementation and the specification. As argued in this paper, this comparison method is often inconclusive because of round-off errors.

Related to DAA algorithms, a comparative analysis between DAIDALUS and ACAS-Xu, another well-known DAA algorithm for UAS, is presented in [4]. In that work, plot diagrams similar to those used in DAA-Displays are employed to present the results of the analysis. The development of analysis tools was, however, not the main focus of their work. While some tools may have been developed, they are not publicly available.

6 Conclusion

A toolkit, called DAA-Displays, has been presented that enables a systematic comparison between DAA software implementations and reference specifications. Because of round-off errors introduced in software implementations, such comparison cannot usually be done by checking numerical differences. The toolkit provides specialized front-end views that enable comparison by simple visual inspection of plot diagrams and interactive display prototypes. Future aspects that will be incorporated into the toolkit include integration with hardware-in-the-loop simulation tools for coupling simulations with hardware modules.

References

1. Carreño, V., Consiglio, M., Muñoz, C.: Analysis and Preliminary Results of a Concept for Detect and Avoid in the Cockpit. In: Proceedings of the 38th Digital Avionics Systems Conference (DASC 2019). San Diego, CA, US (September 2019)
2. Chamberlain, J.P., Consiglio, M.C., Muñoz, C.: DANTi: Detect and Avoid iN The Cockpit. In: 17th AIAA Aviation Technology, Integration, and Operations Conference. p. 4491 (2017). https://doi.org/10.2514/6.2017-4491
3. Cook, S.P., Brooks, D., Cole, R., Hackenberg, D., Raska, V.: Defining Well Clear for Unmanned Aircraft Systems. In: Proceedings of the 2015 AIAA Infotech @ Aerospace Conference. No. AIAA-2015-0481, Kissimmee, Florida (January 2015). https://doi.org/10.2514/6.2015-0481

4. Davies, J.T., Wu, M.G.: Comparative Analysis of ACAS-Xu and DAIDALUS Detect-and-Avoid Systems. Tech. rep. (2018), https://ntrs.nasa.gov/search.jsp?R=20180001564
5. Dutle, A., Muñoz, C., Narkawicz, A., Butler, R.: Software validation via model animation. In: Blanchette, J., Kosmatov, N. (eds.) Proceedings of the 9th International Conference on Tests & Proofs (TAP 2015). Lecture Notes in Computer Science, vol. 9154, pp. 92–108. Springer, L'Aquila, Italy (July 2015). https://doi.org/10.1007/978-3-319-21215-9_6
6. Heitmeyer, C., Kirby, J., Labaw, B., Bharadwaj, R.: SCR: A toolset for specifying and analyzing software requirements. In: International Conference on Computer Aided Verification. pp. 526–531. Springer (1998). https://doi.org/10.1007/BFb0028775
7. Le Sergent, T.: SCADE: A comprehensive framework for critical system and software engineering. In: International SDL Forum. pp. 2–3. Springer (2011). https://doi.org/10.1007/978-3-642-25264-8_2
8. Masci, P., Oladimeji, P., Curzon, P., Thimbleby, H.: Using PVSio-web to Demonstrate Software Issues in Medical User Interfaces. In: Huhn, M., Williams, L. (eds.) Software Engineering in Health Care. pp. 214–221. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-63194-3_14
9. Masci, P., Oladimeji, P., Zhang, Y., Jones, P., Curzon, P., Thimbleby, H.: PVSio-web 2.0: Joining PVS to HCI. In: International Conference on Computer Aided Verification. pp. 470–478. Springer (2015). https://doi.org/10.1007/978-3-319-21690-4_30
10. Muñoz, C., Narkawicz, A., Chamberlain, J., Consiglio, M., Upchurch, J.: A family of well-clear boundary models for the integration of UAS in the NAS. In: Proceedings of the 14th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference. No. AIAA-2014-2412, Georgia, Atlanta, USA (June 2014). https://doi.org/10.2514/6.2014-2412
11. Muñoz, C., Narkawicz, A., Hagen, G., Upchurch, J., Dutle, A., Consiglio, M.: 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). https://doi.org/10.1109/DASC.2015.7311421
12. Narkawicz, A., Muñoz, C., Dutle, A.: The MINERVA software development process. In: Shankar, N., Dutertre, B. (eds.) Automated Formal Methods. Kalpa Publications in Computing, vol. 5, pp. 93–108. EasyChair (2018)
13. Narkawicz, A., Muñoz, C., Dutle, A.: Sensor uncertainty mitigation and dynamic well clear volumes in DAIDALUS. In: Proceedings of the 37th Digital Avionics Systems Conference (DASC 2018). London, England, UK (September 2018)
14. Owre, S., Rushby, J.M., Shankar, N.: PVS: A Prototype Verification System. In: International Conference on Automated Deduction. pp. 748–752. Springer (1992). https://doi.org/10.1007/3-540-55602-8_217
15. RTCA SC-1228: RTCA-DO-365, Minimum Operational Performance Standards for Detect and Avoid (DAA) Systems (May 2017)
16. Titolo, L., Muñoz, C., Feliú, M., Moscato, M.: Eliminating Unstable Tests in Floating-Point Programs. In: Proceedings of the 28th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR 2018. Lecture Notes in Computer Science, vol. 10747, pp. 169–183. Springer (2018)
17. US Code of Federal Regulations: Title 14 Aeronautics and Space; Part 91 General operating and fight rules; Section 111 (1967)
18. US Code of Federal Regulations: Title 14 Aeronautics and Space; Part 91 General operating and fight rules; Section 113 (1967)