Certification Considerations for Adaptive Stress Testing of Airborne Software

Michael Durling
GE Research
Niskayuna, NY, USA
durling@ge.com

Heber Herencia-Zapana
GE Research
Niskayuna, NY, USA
heber.herencia-zapana@ge.com

Baoluo Meng
GE Research
Niskayuna, NY, USA
baoluo.meng@ge.com

Mike Meiners
GE Aviation Systems
Cincinnati, OH, USA
mike.meiners@ge.com

Joachim Hochwarth
GE Aviation Systems
Grand Rapids, MI, USA
joachim.hochwarth@ge.com

Nicholas Visser
GE Aviation Systems
Grand Rapids, MI, USA
nicholas.visser@ge.com

Ritchie Lee
NASA Ames Research Center
Moffett Field, CA, USA
ritchie.lee@nasa.gov

Robert Moss
Stanford Universtiy
Stanford, CA, USA
mossr@stanford.edu

Vidhya Tekken Valapil
GE Research
Niskayuna, NY, USA
vidhya.valapil@ge.com

***Abstract*— Adaptive Stress Testing (AST) has shown promise in identifying errant corner cases in complex software used in aerospace applications including Flight Management Systems (FMS). The strength of AST is performing test-based validation and verification of complex aerospace software-intensive systems at scale in simulated operational environments. Simulating and capturing the realistic operational complexities in integrated verification environments may expose flaws in the software prior to field deployment, whereas the software may perform just fine to traditional requirements-based unit and component-level testing. AST can be used to test code components or the entire system. Individual components may behave safely, but together can result in complex interactions and emergent failures, so it is important to test at the integrated system level. Motivated by the observed benefits at the prototype proof-of-concept scale, this paper considers how AST may be integrated into a production workflow and used to generate objective evidence in a process that delivers certified aerospace software. The research includes evaluation of alignment with both RTCA DO-178C and Overarching Properties (OP). The paper addresses questions such as "where should AST fit in the software development lifecycle, what aspects of AST do not fit, and what objectives does it satisfy?" The paper concludes that AST is useful at locating errors in complex airborne application software and in doing so provides benefits to manufacturers and end users. Furthermore, AST appears to align with in both DO-178C-based and Overarching Properties-based certification approaches.**

Keywords*— *Adaptive Stress Testing, DO-178C, Overarching Properties, software certification, validation & verification

I. INTRODUCTION

The goal for this paper is to consider how Adaptive Stress Testing, which has demonstrated benefits in terms of identifying errors in complex software [1], may be applied in an aerospace software production environment as shown in Figure 1, and how results may be considered as evidence in support of certification. The paper considers both the traditional RTCA DO-178C [2] process and the emerging Overarching Properties (OP) [3] approach. Section II of the paper provides background information on AST, DO-178C and OP. In Section III, potential benefits, and options for alignment with the certification for each of the approaches are described. Section IV shows example implementation artifacts used in the analysis. Section V includes concluding remarks on what was learned and recommendations for next steps.

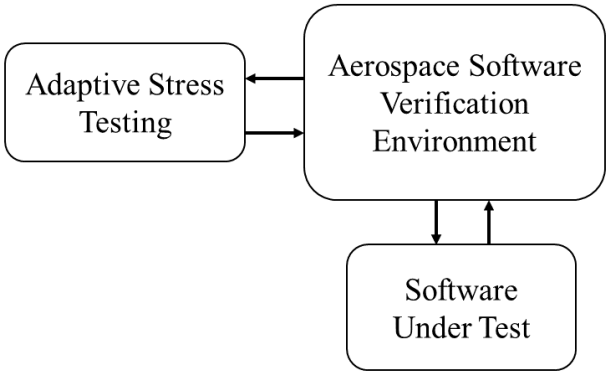


Figure 1: Verification Environment Architecture

II. BACKGROUND

A. Adaptive Stress Testing

Adaptive Stress Testing (AST) is a framework for accelerated validation & verification of safety-critical systems in simulation [4]. AST uses learning to efficiently search the state space for the most likely sequence of states that results in a failure event. The AST framework considers a System Under Test (SUT) interacting with an operating environment. The environment is affected by disturbances that follow a given probability distribution. Rather than sampling values from the disturbance distributions as is done with traditional Monte Carlo testing, AST explicitly optimizes the disturbance values to induce the most likely failures in the SUT. A failure event occurs when the system enters a failed state, which is defined as a subset of the state space.

To perform the optimization, AST formulates stress testing as a sequential decision process and then uses a reinforcement learning algorithm to optimize it. The SUT and the environment are combined into a simulator. At each time step, the reinforcement learning agent observes the state of the simulator, chooses a disturbance, and receives a reward. Through repeated interactions with the simulator, the agent learns to choose disturbances that maximize the reward it receives. The reward function is crafted to reward failure events and higher likelihood transitions, which leads the agent to optimize for the most likely failure path. State-of-the-art reinforcement learning algorithms are leveraged for performance and scalability.

AST has been successfully applied to analyze failure events in many applications, including the next-generation Airborne Collision Avoidance System (ACAS X) [4], autonomous cars [5, 6], trajectory planners in small UAVs [7], autonomous taxiing aircraft [8], and trajectory prediction systems in a developmental Flight Management Systems (FMS) [1].

B. Software certification based on DO-178C

DO-178C "Software Considerations in Airborne Systems and Equipment Certification" [2] is a document published by RTCA in 2011 that provides guidance for development and certification of software used in airborne systems. It serves as the basis for avionics manufacturers to develop and certify software for their products. The document was created by international aviation community working groups. DO-178 was originally developed in the 1980s based on the growth of software content in airborne systems.

DO-178C provides guidance for a process-based approach to software development and certification. Objectives for the software life cycle processes are provided along with activities that may be performed to satisfy them. The document also provides descriptions of evidence that should be generated to show that the objectives have been satisfied. For example, in the verification process there is an objective that states "Test coverage of high-level requirements is achieved". Test cases with expected results must be generated and executed for each requirement, and test result evidence must be captured for each configuration item or software version. Any discrepancy needs to be captured and tracked in a managed problem reporting system. The major software development life cycle processes in DO-178C include software planning, software development, software verification, software configuration management, software quality assurance and certification liaison. The overall process is well documented, very logical and structured to facilitate planning, execution, and review.

C. Software certification Overarching Properties

The DO-178C approach has demonstrated good results, but as software scale and complexity continue to grow, the community is concerned about the cycle time and cost to develop new and update legacy software. In 2015, the FAA initiated research to streamline avionics certification that included development assurance processes which led to a set of Overarching Properties (OP) [3, 4]. In principle, if an entity possesses the three Overarching Properties, 1) Intent, 2) Correctness and 3) Innocuity, then certification approval is considered appropriate. Below are the OP definitions from [3].

1. **Intent:** The defined intended behavior is correct and complete with respect to the desired behavior.
2. **Correctness:** The implementation is correct with respect to its defined intended behavior, under foreseeable operating conditions.
3. **Innocuity:** Any part of the implementation that is not required by the defined intended behavior has no unacceptable impact.

This OP approach is very different than the detailed process specified in DO-178C that the community is accustomed to. For manufacturers of aerospace software to consider using OP, they will need to review detailed examples of the approach in practice, along with an assessment of benefit relative to the traditional method of certification. A good OP example for tool qualification is presented in [5]. This paper aspires to make a small contribution toward helping the community better understand application of OP in an aerospace software development and certification workflow and the potential benefits.

III. APPROACH

AST is designed to guide the failure search using a metric that quantifies "how 'close' is a failure?" This metric, referred to as the *distance* metric, is used in the context of reinforcement learning as the reward signal that we want to maximize, where AST maximizes the negative distance to optimize towards failures. In previous work [1], AST was reformulated as a problem to fit an open-loop SUT (i.e., a system that outputs a single reward signal at the end of a simulation, rather than incremental reward signals at each step of the simulation). As an example, an aircraft trajectory prediction subsystem of a developmental commercial FMS was used as an open-loop SUT to be stressed. Several failure modes of the SUT were proposed with the main failure being inconsistencies in the predicted lateral trajectories. More specifically, the failure mode was defined by differences in the predicted arc length of a turn segment compared to the calculated arc length from the angular extent and arc radius

output values. Differences in these quantities manifest themselves as disconnections between incoming and outgoing waypoints across turn segments, resulting in errors in the predicted trajectory. A simulator was built around AST to sample waypoints and environment conditions such as at-altitude wind, and AST controls samples within the simulator to intelligently select full sets of trajectories that likely resulted in a failure. Therefore, AST passed a candidate set of waypoints to the SUT and then parsed the miss distance (i.e., the predicted vs. calculated arc length difference) from the SUT output. Given assumptions on the initial conditions within the simulator and the sampling distributions of the environment, prior experiments empirically showed AST was able to find failures in about 88% of system executions while other benchmark approaches found failures in less than 1% of system executions. The failures found by AST were broadly classified into two types of failure modes: those with duplicate sequential waypoints and those with near-duplicate sequential waypoints. That is, cases where the same waypoint (or very close in position) were selected one after another often resulted in a trajectory failure. These discovered failure modes provided the engineers and developers many example cases which helped to identify potential problems in the SUT itself. Because the simulation environment is continuous in nature, the AST failure search has an infinite search space to find failures. What may seem like a daunting limitation actually provides a benefit. Running AST as a supplemental test in addition to other requirement-based testing suites allows the developers to potentially find failure modes otherwise not found in hand-crafted test cases.

Production test suites or testing environments can benefit by integrating AST into them as an additional testing tool or capability. AST can improve the ability of a test suite in identifying software failures that are hard to detect using traditional testing approaches. However, challenge lies in the fact that the AST architecture involves continuous direct invocation of the system under test (SUT) with test inputs and direct access to the SUT output to determine the reward information, which in turn helps in identifying the next set of inputs. Therefore, to enable integration of AST into an existing test suite, one has to modify the continuous workflow of AST in such a manner that the test suite continues to be the controlling entity. Modifying the workflow would involve making the test suite obtain test inputs from AST, then execute the SUT with the inputs, and based on the output of SUT make the reward information available to AST to help determine the next test input. With the modified workflow the test suite will continue to act as the entity in charge, while also enabling the AST-SUT interaction that is key to AST.

Figure 2 shows the flow of hazards captured as error conditions along with their failure severity level from the safety assessment to the system process then the software process to software verification. At the system level, the hazards are allocated to functional requirements that are allocated to software along with the failure severity level that maps, in DO-178C, to Development Assurance Levels (DAL). At the software level, the high-level requirements (HLR) are refined into low-level requirements (LLR) then implemented as source code. During the software verification process, AST may be used to verify that error conditions identified in the safety assessment are not present in the software implementation. DO-178C provides clear guidance in terms of what objectives, including testing needs to be performed to satisfy each DAL. OP [3] includes a constraint that states "The means by which the defined intended behavior is shown to be correct and complete is commensurate with the DAL" but does not specifically map the DAL to DO-178C objectives. OP leaves it up to the user to define what needs to be completed to satisfy each DAL. Figure 2 also shows the alignment of Overarching Properties with the traditional development process.

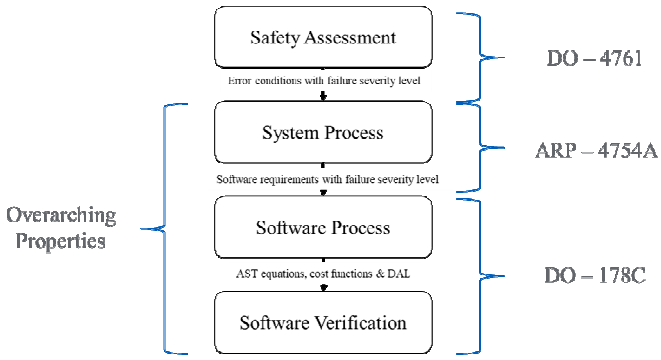


Figure 2: Flow from Safety Assessment to AST

A. *Expected Benefit Hypothesis for AST*

DO-178C requirements-based testing focuses on test cases being developed against requirements. These test cases only cover certain conditions (e.g. in-range and out-of-range inputs), but not necessarily every possible complex input. They do, per DO-178C, fully test the requirements as stated. However, the overall system is more complex than that and is also working in a highly volatile environment of real-world data – for instance, data that is updated every 28 days via ever-changing navigational data (Aeronautical Information Regulation and Control (ARIAC)). This complexity makes it very difficult to consider every possible input/test case in requirements-based testing.

Adaptive Stress Testing is designed to perform testing on systems that have any of two properties: 1) highly complex systems or collections of systems that have to work together, and 2) systems that have a large number of inputs or inputs that can take on non-discrete, continuous values. The former condition implies that it is difficult for the requirements-based tester to know what outcome to expect from the input of the complex system, and the latter implies that any method of requirements-based testing will only just scratch the surface of possible variations of inputs to the system.

B. *Alignment to DO-178C*

DO-178 is a proven standard for developing safe software for airborne systems. As systems become more complex and software intensive, the prescriptive "requirements-based" focused standard has been showing its age with respect to allowing for non-traditional software development and verification methodologies. There are three possible ways an applicant could propose to use AST within the confines of the

guidance in DO-178. These would be subject to acceptance by the project's airworthiness authority. These three use cases are Robustness Testing, Exhaustive Testing, and Product Service History.

Robustness Test Cases in DO-178 are used to establish the software's ability to respond correctly to abnormal inputs and conditions. Unlike normal test cases, the development of these robustness test cases can be difficult as it is not always easy to identify all abnormal inputs and conditions, especially for complex systems. AST may be useful in these cases as AST excels at testing software at scale. The technology can be used to drive robustness into the software. As the testing find errors, the related requirements, design, and code are corrected. While this is a valuable activity, these errors may not be strictly due to abnormal inputs or conditions, nor can they be considered requirements based (DO-178C added a note that robustness test cases are to be requirements-based), meaning no credit can be taken under DO-178. Outside the informal use of AST to increase the maturity of the software, value could be claimed by identifying test cases from errors that AST finds that are due to abnormal inputs or conditions, then captured as DO-178 robustness test cases.

Exhaustive Input Testing is an established alternative method for verification recognized by DO-178. While useful, its applicability is limited by the ability to bound the set of inputs and outputs of the given software component. This limits the practical application of exhaustive input testing to only simple software logic. For more complex software systems, AST could be proposed as a means to incrementally approach the full bounds of a larger input space, enabling the claim of exhaustive-enough testing of complex software logic.

Product Service History is another alternative method described in DO-178. It allows for some certification credit to be obtained through the demonstration of equivalent safety from the product's service history data. DO-178 contains guidance on the acceptability of this method. This guidance would still apply when using AST. What would change is instead of using historical product service history data, simulated data would be used instead. Simulation of software in the target environment using test input is not new. However, AST would be an enabling technology to quickly accumulate virtual service hours using real-world test inputs from sources such as FlightAware. Using FlightAware planned and actual data is ideal for evaluating software under "foreseeable operating conditions". This product service data would then be applied not to a new use of existing fielded software, but for the verification credit of new software proposed for approval.

C. *Alignment to Overarching Properties*

The OP aim to provide an alternative path for approval. The OP path is intended to be more abstract and less prescriptive, and thus allow greater flexibility in showing design assurance. Currently there is a desire for more details about how the OP might be used in practice, especially how to show and evaluate whether a product possesses the OP. In related research [5] is a good reference for using OP for tool qualification. This section shows how AST users may apply OP to argue for certification value of testing artifacts. This is done in three steps. The first step, **requisites fulfillment**, consists of checking that the six requisites from [3] are fulfilled. For example, it needs to present what is the desired behavior and the Defined Intent Behavior (DIB). The second step is **assumption fulfillment**. This step consists of checking that the assumptions from [3] are fulfilled. For example, it needs to be assumed that the stakeholder knows the desired behavior for the software component. The third step, **OP possession and Goal Structuring Notation** (GSN) shows how to generate the OP properties and how to show the possession of the OP properties. For example, the Intent property for the software component could be generated instantiating the Intent definition using the DIB and the desired behavior. Then guide a demonstration of the Intent property using the Constraints. Finally, a GSN assurance case is generated for the demonstration of the OP possessions.

IV. IMPLEMENTATION

A. *DO-178C Artifacts*

Below, an example method of using AST in the DO-178C framework to design requirements, test cases, and test results that leverage the AST tool is presented. The first step is to design a requirement. Generally, AST is used to find failure events in complex systems. These failure events should not occur in the system, and thus a requirement can be drafted using the "shall not" language, as shown in Figure 3. An equivalent requirement is also shown expressed with traditional "shall' language.

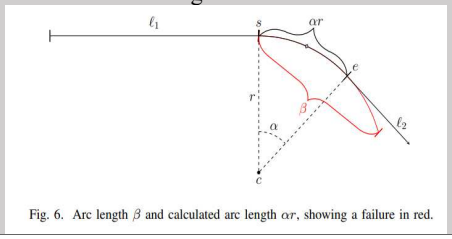
ID	Requirement	In-Links
HLR-1234	For FMS-produced lateral path curve segments, the difference between the listed arc length and the arc length computed using the angular extent and arc radius ***shall not*** be greater than 10 ft. Alternatively, this requirement could be: For FMS-produced lateral path curve segments, the difference between the listed arc length and the arc length computed using the angular extent and arc radius ***shall*** be less than or equal to 10 ft.	TC-1234
HLR-1235	[COM] The diagram below shows the arc length check for a curve segment:  Fig. 6. Arc length β and calculated arc length αr, showing a failure in red.	None

Figure 3: DO-178 Style Requirement

Once the requirement is drafted, a test case can be linked to this requirement, demonstrating to the tester the operating conditions and scenario for verifying the requirement in

testing. In traditional requirements-based testing, this process involves drafting a test case describing the testing conditions, and a test procedure with specific instructions to the tester on how to accomplish the test. The test procedure is linked to the test case, and the test case to the requirement. Examples of these are shown in Figures 4 and 5.

Once the test case is drafted and some general instructions on how to use the AST tool are disseminated, a tester has everything needed to verify the requirement per DO-178C. If the test procedure is run as described and the arc length failure event is not discovered, then the test case has passed. A log detailing when the test was run, what build was used, and the passing result can be generated. The log would enable the test to be reproduced. In the event of a failure (or failures) found by AST, the details related to the failure can be logged, including the specific sample of waypoints and winds that created the test case resulting in failure. These details can be included in a problem report database or tracker, allowing a developer to recreate the issue to resolve it, and run the test case again. In the event that more than one failure is found, the failures can be ranked according to their likelihood, as determined by AST.

ID	Test Case	Out-Links	In-Links
TC-1234	***Verify*** that Adaptive Stress Testing is unable to create a set of test inputs that lead the FMS Trajectory Predictor to generate an arc length (β) that differs from the arc length computed using the angular extent (α) and arc radius (r) by more than 10 feet, when: - The origin airport is KSFO - The destination airport is KLAX - The distribution for waypoint direction is normal with mean of 180 degrees and variance of 45 degrees - The distribution for waypoint distance is normal with mean of 50 nautical miles and variance of 30 nautical miles - The distribution for wind direction is normal with mean of -88.5 degrees and variance of 39.5 degrees - The distribution for wind magnitude is normal with mean of 66.8 knots and variance of 24.4 knots - The number of iterations is limited to 5,000	HLR-1234	TP-1234

Figure 4: DO-178 Style Test Case

ID	Test Procedure	Out-Links
TP-1234	1. Load the AST tool 2. Select the "4-D Trajectory Predictions" Domain 3. Enter KSFO as the origin airport 4. Enter KLAX as the destination airport 5. Enter "normal" as the distribution for waypoint direction, with mean of 180 degrees and variance of 45 degrees 6. Enter "normal" as the distribution for waypoint distance, with mean of 50 nautical miles and variance of 30 nautical miles 7. Enter "normal" as the distribution for wind direction, with mean of -88.5 degrees and variance of 39.5 degrees 8. Enter "normal" as the distribution for wind magnitude, with mean of 66.8 knots and variance of 24.4 knots 9. Enter 5,000 as the number of iterations 10. Run the AST tool 11. Once the run has completed, ***verify*** that AST indicates "pass"	HLR-1234

Figure 5: DO-178 Style Test Procedure

B. OP Artifacts

Requisites fulfillment. Requisites must exist to allow OP possession to be shown. They do not constrain how the demonstration must be done but establish preconditions that must be true before a successful demonstration of property possession is possible [3]. Let us illustrate this step with the generation of the following requisite "Defined intended behavior exists" for the Trajectory Predictor product. The Desired behavior encompasses everything the stakeholders want the product to do, along with anything that they want to ensure it does not do. For our example we are going to focus on what the system should not do. Then, the desired behavior is "The Trajectory Predictor shall not produce waypoints which are unreachable given the physical limitations of the aircraft" and this desired behavior is called No unreachable Waypoints. The Defined Intended Behavior (DIB) is a representation (that is, a record) of the intellectual understanding of the desired behavior. For our example the DIB focuses on arc length discrepancies as a primary source of unreachable waypoints and is stated as "The FMS-produced lateral path *shall not* contain a curve segment where the listed arc length is different than the arc length computed using the angular extent and arc radius by more than 10 feet." and this DIB is identified as Req-ID HLR-1234. Table 1 shows the requisite fulfillment.

TABLE 1: THE SIX OP REQUISITES

Requisites	Requisites for the Trajectory Predictor
a. Defined intended behavior exists.	The FMS-produced lateral path ***shall not*** contain a curve segment where the listed arc length is different than the arc length computed using the angular extent and arc radius by more than 10 feet. (REQ-ID HLR-1234)
b. Failure conditions are defined.	Arc Length Failure
c. The record of the safety assessment exists.	A safety assessment should be used to identify the error conditions evaluated by AST.
d. The record of the foreseeable operating conditions exists.	Inputs of the simulation are the winds aloft, origin and destination airports, and the placement of the lateral waypoints.
e. The implementation exists.	Trajectory Predictor code
f. Design Assurance Levels (DALs) are assigned using the failure condition classifications.	*No Unreachable Waypoints* require DAL B (the team assumed that the consequence of the anomalous behavior as shown by the system safety assessment process would cause or contribute to a hazardous failure condition for the aircraft.)

Assumption fulfillment. Assumptions are only stated, not justified, in the demonstration of the possession of the Overarching Properties. Table 2 shows the assumptions for the Trajectory Predictor.

TABLE 2: THE ASSUMPTIONS

Assumptions	Assumptions for the Trajectory Predictor
a. Stakeholders have the knowledge to express the desired behavior	The trajectory predictor shall not produce waypoints which are unreachable given the physical limitations of the aircraft
b. Performing safety assessment is not covered by these Overarching Properties	A safety assessment was performed and identified the error conditions used by AST.

OP possession and GSN. A demonstration of a statement needs to have some facts and state some arguments that follow from the given facts; then some arguments that follow from those arguments; and so on until the statement is reached. For the demonstrations of the overarching properties statements, the facts and arguments are guided and generated by the OP Constraints [3], because the constraints apply directly and only to how OP possession may be demonstrated. That is, they constrain what is considered a legitimate demonstration. Having said that, this section shows how the demonstration that the Trajectory Predictor has the Overarching Properties is guided by the Constraints.

Intent: The users need to demonstrate that The DIB "Req-ID HLR-1234" is correct and complete with respect to the desired behavior "No Unreachable Waypoints". To illustrate the demonstration of this property let us first capture some facts using the constraint C.a: "The process to ensure possession of the Overarching Properties must be defined and conducted as defined" [3]. This constraint allows the user to define the processes that will be used and follow those processes once they are defined. For this example, the DAL in the sense of DO-178C was considered in order to drive the demonstration of the possession of this Intent property. Now having this fact, construct the main argument of the demonstration which is suggested by the constraint C.b: "The means by which the DIB is shown to be correct and complete is commensurate with the DAL" [3]. This constraint guides and allows use of the product's DAL for the possession of the intent property as follows: The DIB Req-ID HLR-1234 requires DAL B, because the consequence of failure of the desired behavior *No Unreachable Waypoints* is assumed to be hazardous. Table 3 summarizes the main points of this demonstration and Figure 6 shows the same information in GSN view.

Table 3: Intent Goals & Strategy

Goal 1	The DIB "Req-ID HLR-1234" is correct and complete with respect to the desired behavior "*No Unreachable Waypoints*".
Strategy	The constraints drive the means by which Intent possession is demonstrated.
Goal 2	The DAL in the sense of DO-178C is considered in order to drive the demonstration of the possession of this Intent property. This statement complies with constraint C.a Solution: According to constraint C.a, the applicant defines the processes that will be used. For this case it is DAL in the sense of DO-178C
Goal 3	Constraint C.b guides the sentence - the DIB Req-ID HLR-1234 is assigned DAL B, because the consequence of failure of the desired behavior *No Unreachable Waypoints* is hazardous. Solution: C.b constraint explicitly allows for different means to be used to show possession of the Intent property depending on the product's DAL.

Correctness: The implementation of the Trajectory Predictor is correct with respect to its DIB "Req-ID HLR-1234", under foreseeable operating conditions. To show the possession of this property it is known that satisfying C.b should result in DAL-commensurate activities being applied for those OP, such as Correctness. This means that constraint C.b guides the following statement for the evaluation and

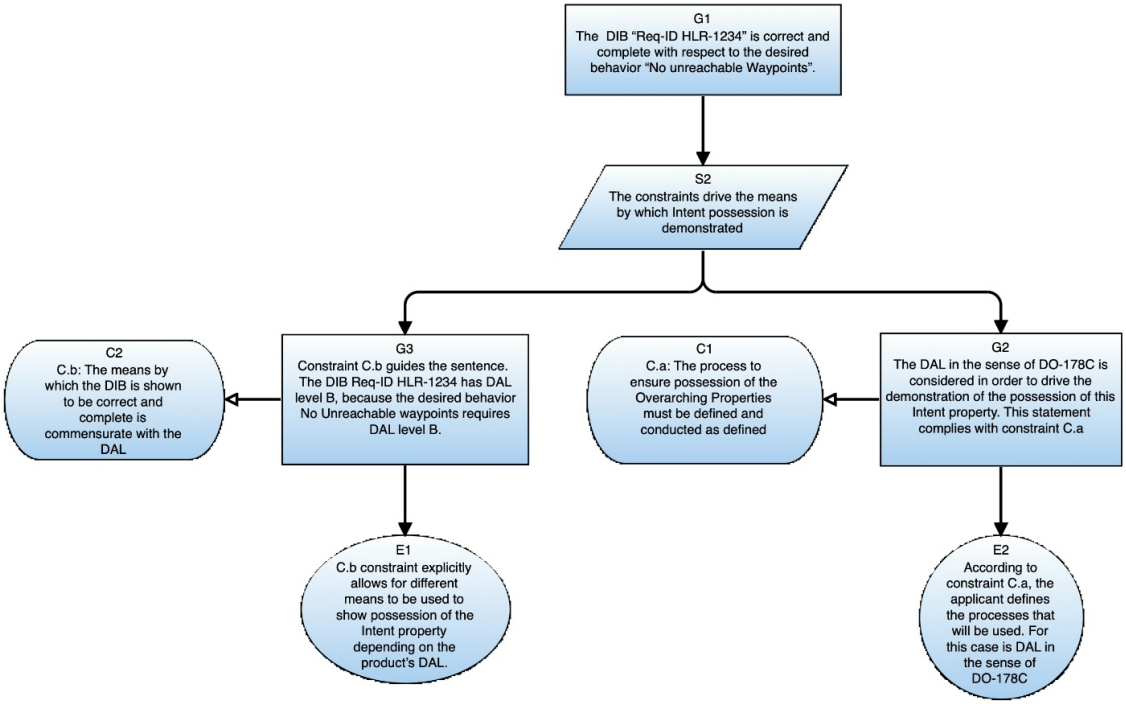


Figure 6: GSN for Intent Property

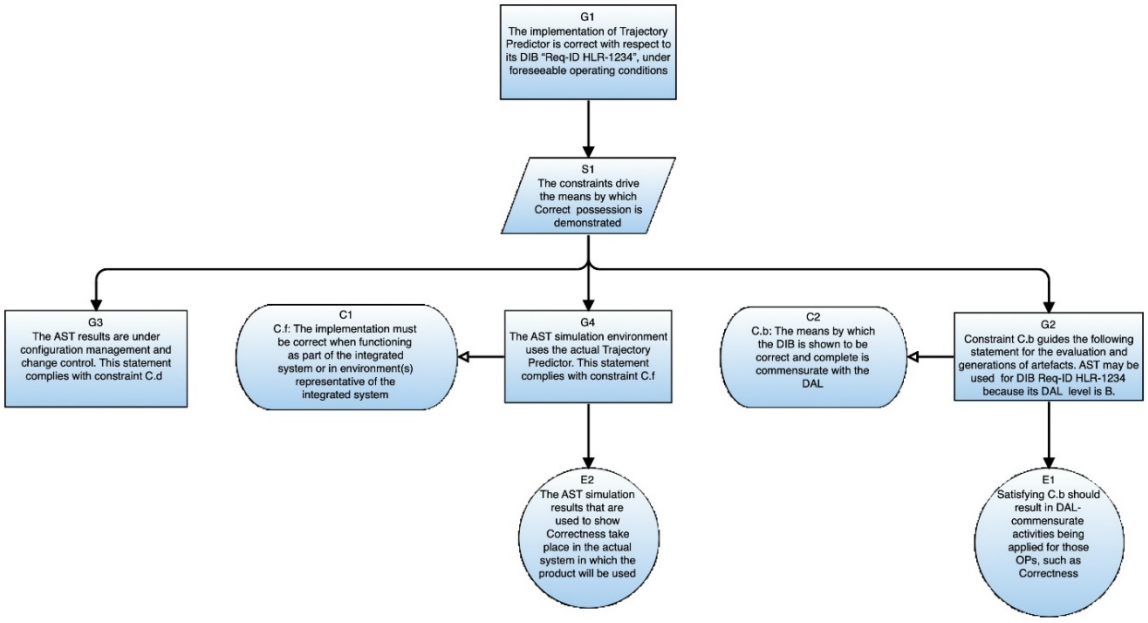


Figure 7: GSN for Correctness Property

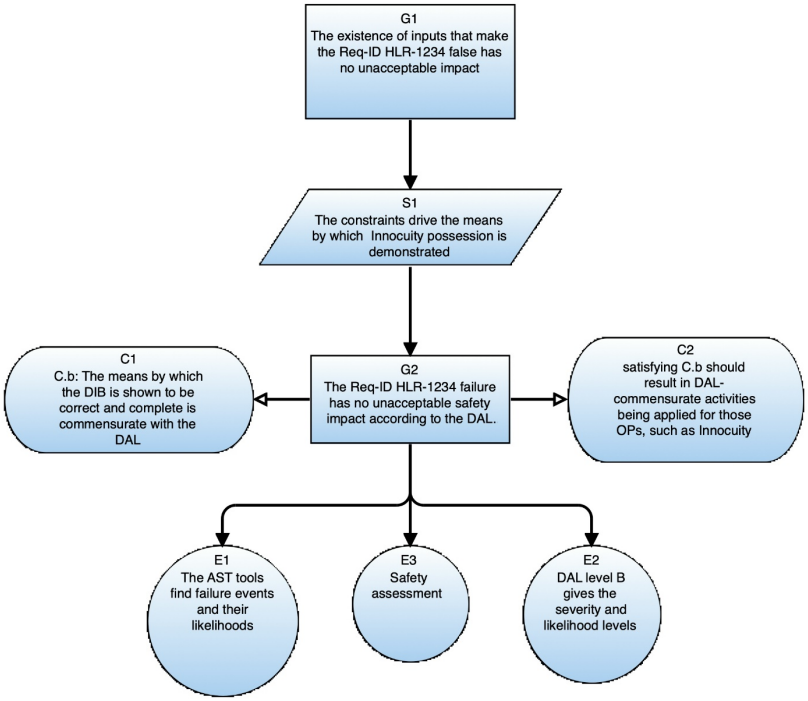


Figure 8: GSN Innocuity Property

generation of artifacts. AST may be used to show Correctness for Req-ID HLR-1234. This means that AST will generate and evaluate artifacts for the correctness of the Trajectory Predictor with respect to the DIB Req-ID HLR-1234 under foreseeable operating conditions. But the foreseeable operating conditions need to be defined. The constraint C.f "The implementation must be correct when functioning as part of the integrated system or in environment(s) representative of the integrated system" [3] helps to define the foreseeable conditions. This constraint exists to ensure that demonstrations of Correctness take place in either the actual system in which the product will be used, or in one or more environments that represent the actual system in all relevant aspects. For AST, a simulation environment was constructed that uses the actual Trajectory Predictor, where the main inputs of the simulation are the winds aloft, origin and destination airports, and the placement of the lateral waypoints. After running the AST tool, there are two possible outcomes. First, if the tool only finds a set of inputs that make the Req-ID HLR-1234 true, then these results are presented as artifacts and/or evidence of the Correctness of DIB Req-ID HLR-1234. Second, if the tool finds a set of inputs that make the DIB Req-ID HLR-1234 false and it is not possible to fix or it is not worth the effort, then the AST tool results need to be incorporated into the Innocuity property. Table 4 summarize the main points of this demonstration and Figure 7 shows the same information in GSN view.

TABLE 4: CORRECTNESS GOALS & STRATEGY

Goal	The implementation of the Trajectory Predictor is correct with respect to its DIB "Req-ID HLR-1234", under foreseeable operating conditions
Strategy	The constraints drive the means by which correct possession is demonstrated.
Goal 1	The AST simulation environment uses the actual Trajectory Predictor. This statement complies with constraint C.f Solution: The AST simulation results that are used to show Correctness take place in the actual system in which the product will be used.
Goal 2	Constraint C.b guides the following statement for the evaluation and generations of artifacts. AST may be used for DIB Req-ID HLR-1234 because its DAL B. Solution: Satisfying C.b should result in DAL-commensurate activities being applied for those OPs, such as Correctness.
Goal 3	The AST results are under configuration management and change control. This statement complies with constraint C.d

Innocuity: The existence of inputs that make the Req-ID HLR-1234 false has no unacceptable impact. To show the possession of this property it is known that satisfying C.b should result in DAL-commensurate activities being applied for those OP, such as Innocuity. So, there is a need to demonstrate that Req-ID HLR-1234 failure has no unacceptable safety impact according to DAL B. A demonstration of the innocuity property according to DAL B is constructed using the likelihoods, severity, and connected back

to the safety assessment. Table 5 summarizes the main points of this demonstration and Figure 8 shows the same information in GSN representation.

TABLE 5: INNOCUITY GOALS & STRATEGY

Goal 1	The existence of inputs that make the Req-ID HLR-1234 false has no unacceptable impact
Strategy	The constraints drive the means by which Innocuity possession is demonstrated
Goal 2	The Req-ID HLR-1234 failure has no unacceptable safety impact for DAL B
Solutions	Safety assessment, the AST tool finds failure events and their likelihoods.

V. CONCLUSION

This paper considered certification implications for use of AST in a production workflow for aerospace software. In 2020, [1] showed that AST was useful in identifying errors in complex software. While performing this work, the team identified additional applications for AST that include identification of worst-case execution time (WCET) and difficult to find structural coverage test cases. The primary goal for this project was to explore opportunities for use of AST to produce DO-178C or OP certification artifacts and evidence. For DO-178C, robustness and exhaustive input testing and product service history were identified and described in this paper as potential alignment opportunities. For an OP based certification approach, concepts were presented that showed alignment with intent, correctness and innocuity properties.

After considering both the DO-178C and OP based concepts the research team concluded that AST can provide value to the aerospace software validation and verification process by locating difficult to find errors while producing useful evidence for certification. The DO-178C evidence was clearly aligned with specific objectives. The OP conceptual alignment was clear, in fact the AST test results are recommended as evidence for the correctness and innocuity properties. When the team came to the point that they needed to specify required evidence to satisfy a specific DAL for an OP, they initially pointed to DO-178C objectives that call for specific artifacts for each DAL. However, it seems that if OP is dependent upon DO-178C for specifying objectives required for each DAL, then it does not simplify the process, it just adds another layer. After discussions the team concluded and recommends that more work is needed to come to agreement of what specific evidence is appropriate to satisfy OP for each DAL. The OP recommendation applies to AST but is generally applicable for any software development and certification.

ACKNOWLEDGMENT

The authors thank James Lopez and Gary Quackenbush for initiating this activity and Guillaume Brat and Paul Miner for their technical advice. This work was supported by NASA SWS Grant #80NSSC19M0239.

REFERENCES

[1] Robert J. Moss, Ritchie Lee, Nicholas Visser, Joachim Hochwarth, James G. Lopez, Michael J. Kockenderfer, "Adaptive Stress Testing of Trajectory Predictions in Flight Management Systems," in *IEEE DASC* , San Antonio, Texas, 2020.

[2] "DO-178C: Software Considerations in Airborne Systems and Equipment Certification," RTCA, Washington, D.C., 2011.

[3] C Michael Holloway , "Understanding the Overarching Properties," NASA Technical Memorandum 20190029284, Hampton, Virginia, 2019.

[4] James Chelini, Jean Camus, Cyrille Comar, Duncan Brown, Anne-Perrine, "Avionics Certification: Back to Fundamentals with Overarching Properties," in *Embedded Real Time Systems ERTS 2018*, Toulouse, France, 2018.

[5] M Anthony Aiello, Cyrille Comar, Jose Ruiz, "An Assurance Case based on Overarching Properties for a TQL1 Code Generator," in Embedded Real Time Systems ERTS 2020, Toulouse, France, 2020.

[6] Lee, Ritchie and Mengshoel, Ole J. and Saksena, Anshu and Gardner, Ryan and Genin, Daniel and Silbermann, Joshua and Owen, Michael and Kochenderfer, Mykel J., "Adaptive Stress Testing: Finding Likely Failure Events with Reinforcement Learning," Journal of Artificial Intelligence Research, vol. 69, pp. 1165--1201, 2020.

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8] Lee, Ritchie and Mengshoel, Ole J. and Agogino, Adrian K. and Giannakopoulou, Dimitra and Kochenderfer, Mykel J., "Adaptive Stress Testing of Trajectory Planning Systems," in AIAA SciTech, Intelligent Systems Conference (IS), 2019.

[9] Kyle D. Julian and Ritchie Lee and Mykel J. Kochenderfer, "Validation of Image-Based Neural Network Controllers Through Adaptive Stress Testing," in IEEE International Conference on Intelligent Transportation Systems (ITSC), 2020.

[10] Anthony Corso and Peter Du and Katherine Driggs-Campbell and Mykel J. Kochenderfer, "Adaptive Stress Testing with Reward Augmentation for Autonomous Vehicle Validation," in IEEE International Conference on Intelligent Transportation Systems (ITSC), 2019.