Leveling Arguments: Easier Said Than Done

Zamira Daw
Raytheon Technologies Research Center
Berkeley, California, USA
zamira.daw@rtx.com

Scott Beecher
Pratt & Whitney
East Hartford, Connecticut, USA
scott.beecher@prattwhitney.com

Michael Holloway
NASA Langley Research Center
Hampton, Virginia
c.michael.holloway@nasa.gov

***Abstract*— The aerospace industry's current desire to rapidly adopt new technologies is inconsistent with the traditional approach to creating new aerospace certification standards. To address this, an FAA-sponsored international Overarching Properties Working Group comprising industry representatives and certifying agencies has introduced the concept of Overarching Properties (OPs). OPs provide a foundation for proposing alternative Means of Compliance (MoC) by demonstrating a product's possession of the Intent, Correctness, and Innocuity properties. Handling criticality levels with the OPs is in its infancy but still required to integrate alternative MoC into the existing aerospace certification infrastructure. This paper examines the impact of criticality levels on the creation and evaluation of OP-related arguments (OPRAs), specifically focusing on the Auxiliary Power Unit (APU) system. The study highlights the need for tailored OPRA creation based on assigned criticality levels and proposes self-assessment techniques to strengthen arguments. This research serves as a stepping stone in exploring criticality levels for OPs and bridging the gap in aerospace certification practices.**

Keywords—assurance, argument, approval, safety, criticality, design assurance levels, overarching properties

I. Introduction

The aerospace industry's current desire to rapidly adopt new technologies is inconsistent with the traditional approach to creating new aerospace certification standards. Some of these new methods and technologies are becoming prevalent in unmanned aerial systems (UAS), hybrid-electric propulsion, and other aerospace innovations. These include methods using artificial "intelligence" / machine "learning" (AI/ML) and agile development methods. The Overarching Properties (OPs) [1] provide a foundation allowing the proposal of alternative Means of Compliance (MoC) based on showing a product possesses three specific properties, which are labeled Intent, Correctness, and Innocuity. These properties give a lot of flexibility to the applicants in deciding what technologies and methods to use for development and assurance, while providing guidance about how to ensure the resulting products satisfy the applicable regulations and policies. These two features (flexibility and guidance) might be the key ingredients required to streamline the certification for new technologies.

This research was supported by NASA through grant 80NSSC20M0006.

In recent years, there have been initial efforts to showcase the application of OPs. These efforts encompass the possession of OPs for an automatic code generator [3], the evaluation criteria for micro-UAV [4], the assessment of Commercial Off-The-Shelf (COTS) components [5], the validation of safety net technology for UAS [6], the usage of adaptive stress testing [7], an on-board UAV model [8], and an auxiliary power unit developed via model-based system engineering [9]. However, the majority of these papers do not address criticality or only mention the level of the target system [6,7,8,9]. An initial discussion on the limitations of guidance on criticality when employing OPs is presented in [7]. Specifically, a comparison is drawn with the objectives of DO-178C, providing a surrogate for criticality levels. While this comparison can serve as a valuable starting point, it is important to recognize that it may not be applicable to technologies that lack support from established standards, which is a primary focus for OPs.

All these efforts have focused on demonstrating OP possession for a single, pre-determined criticality level. Such a focus is understandable because no guidance has yet been published concerning how to handle criticality levels within the context of OPs. The notion of applying criticality levels to the development and assurance of aviation systems has been around for decades (ARP 4761). In contrast, the notion of the OPs is less than seven years old, and OP-Related Arguments (OPRAs) barely over one [2]. Leveling is a mature adult, OPs a young child, and OPRAs a toddler. In today's aviation world, neither toddler nor child will be able to grow much older unless they are able to get along well with the adult. In light of this, our paper seeks to bridge the existing gap by presenting our practical experiences in applying OPs across different criticality levels, and sharing valuable insights gained while examining the effect of criticality level on the creation and evaluation of an OPRA

We begin by providing a bit of history and current state of the art and practice concerning the OPs, OPRAs, and the use of criticality levels in the aviation industry. We then explain current practices for assigning activities and objectives according to design assurance levels (DALs), followed by some background on OPRAs. Afterward, we introduce our industrial case study, present OPRAs developed for two different DALs, and explain why our results suggest the current leveling approach is inadequate for OP-based approval. We conclude by proposing a different but compatible self-evaluation method to help applicants and evaluators to reason about leveling using OPs.

II. Background

A. History of the Overarching Properties

The Overarching Properties originated in a by-invitation-only workshop in December 2015 sponsored by the Federal Aviation Administration (FAA). The FAA selected the invitees to ensure industry and governmental participation from across a wide area of technical disciplines, countries, and assurance viewpoints. The workshop was intended to generalize existing assurance objectives for software, hardware, and systems containing them into a small set of what were initially called ``meta-objectives." The effort continued with two more invitation-only in-person meetings in April and July 2016, periodic virtual meetings, and the use of an online forum, culminating in a set of three, as they were now more appropriately named, Overarching Properties.

These OPs were presented to the public at the 2016 FAA Streamlining Assurance Processes Workshop, which was held September 13--15, 2016, in Richardson, Texas. Workshop participants expressed opinions across a wide spectrum ranging from deliriously enthusiastic to decidedly miffed. By far the larger number (and percentage) of opinions were positive; the work continued.

In early 2017 the team was named the Overarching Properties Working Group (OPWG). Efforts in 2017 and 2018 involved trying to develop a collection of criteria for use in evaluating OPs possession, with the hope of publishing a single document describing the OPs and the possession criteria. However, the results of several case studies completed in 2018 identified deficiencies in criteria-based approaches.

Consequently, the OPWG decided to publish a description of the Overarching Properties alone. A few minor changes were made during a meeting of the group in April 2019, leading a few months later to the publication of [1]. The document introduced the OPs as constituting a sufficient set of properties upon which approval decisions may be made. That is, if an entity (for example, a hardware or software subsystem) for which approval is sought can be shown to possess these properties in their entirety, then granting approval for using that entity on an aircraft is technically appropriate. The anticipated most likely use for the OPs was (and continues to be) expected to consist of subsystems using novel approaches that are not good fits for existing guidance.

Over the next several years the OPWG explored alternatives to the unviable (criteria-based) approach for evaluating whether a particular system possesses the OPs. The group eventually settled on an argument-based approach, the details of which were published in early 2022 [2]. That paper introduced an approach to demonstrating and assessing property possession based on using explicit arguments, which is the approach we use, and about which we will explain much more shortly.

B. OPs & Overarching Properties Related Argument (OPRA)

The OPs were developed from the distillation of existing standards and research-based contemplation about what is needed for complete assurance. The defining document distinguishes among labeled property statements, definitions, requisites, assumptions, and constraints. The meaning of the properties is fully specified by the three property statements (not including the labels) and eight definitions. The labeled property statements are as follows, with small caps denoting phrases for which specific definitions are provided:

- ***Intent:*** The DEFINED INTENDED BEHAVIOR is correct and complete with respect to the DESIRED BEHAVIOR.
- ***Correctness:*** The IMPLEMENTATION is correct with respect to its DEFINED INTENDED BEHAVIOR, under FORESEEABLE OPERATING CONDITIONS.
- ***Innocuity:*** Any part of the IMPLEMENTATION that is not required by the DEFINED INTENDED BEHAVIOR has no UNACCEPTABLE IMPACT.

These property statements cannot be understood without knowing the definitions; thus, they are reproduced here:

a. DESIRED BEHAVIOR: Needs and constraints expressed by the stakeholders (this includes those needs and constraints identified by the SAFETY ASSESSMENT and those mandated by regulations). (may be abbreviated as DeB)

b. DEFINED INTENDED BEHAVIOR: The record of the DESIRED BEHAVIOR. (may be abbreviated as DIB)

c. IMPLEMENTATION: ITEM or combination of inter-related ITEMS for which acceptance or approval is being sought.

d. ITEM: A hardware or software element having bounded and well-defined interfaces.

e. FORESEEABLE OPERATING CONDITIONS: External and internal conditions in which the system is used, encompassing all known normal and abnormal conditions.

f. UNACCEPTABLE IMPACT: An impact that compromises the SAFETY ASSESSMENT.

g. SAFETY ASSESSMENT: The systematic identification of FAILURE CONDITIONS and classifications in an operational context, evaluation of the architecture against safety objectives arising from these hazards, evaluation of potential common modes and threats, defining additional intended behaviors to support claims within these evaluations and showing that the safety objectives are satisfied by the implementation.

h. FAILURE CONDITION: "A condition having an effect on the [aircraft] and/or its occupants, either direct or consequential, which is caused or contributed to by one or more failures or errors, considering flight phase and relevant adverse operational or environmental conditions or external events." [13]

The statements and definitions fully define the OPs. Because the remaining sections of the description—requisites, assumptions, and constraints—only affect the means by which property possession may be shown, they are not shown here. The interested reader should consult [1].

Applicants can use OPs as a foundation on which to seek software, hardware, and system approvals. We use the hybrid approach proposed in [8] for creating OPRAs. The hybrid approach uses existing standards where they are effective, and OPs for proposing alternative MoC. Applicants can define the scope of OPRA either by attaching the OPRA to conventional certification process documents (e.g., Plan for Software Aspects

of Certification — PSAC) or by supporting the OPRA with evidence acquired by existing certification standards (e.g., DO-330 for tool qualification).

Structured arguments are used to state the rationale that an applicant's proposal for demonstrating compliance possesses the OPs [6,8,9]. In our experience, structured arguments facilitate the creation of such rationale by the applicants and its evaluation by the certification authorities. There are many notations [10-12] one could use to represent an OPRA. As proposed in [8], we use a combination of textual and graphical notations to present an OPRA. The textual representation helped us to focus on the details of the argument without distraction of the structure and the graphical notation facilitates the understanding of the big picture and the relationship between atomic arguments for folks for which graphical notations do such a thing.

The Friendly Argument Notation (FAN) [10] is used as textual notation based on a small number of terms easy to understand. Based on our experience, FAN facilitates the use for assurance arguments by ordinary engineers without computer science knowledge. FAN rests on the following concepts: *Argument* is an attempt to convince others to believe a conclusion through reasoning and one or more premises; *Believe* is accepted as true; *Premise* is a statement you think your audience believes; *Reasoning* states why you think the premises should cause your audience to believe your conclusion; *Binding* is an association between a term used in an argument and the real-world information to which that term refers; *Defeater* is a statement that may cause your audience to not believe your conclusion; *Conclusion* is the statement you want your audience to believe.

Believing
Subsystem SAM and IAM both possess Innocuity {IO-Sys}
Is justified by applying
the principle of conjunction
To these premises
1. SAM possesses Innocuity {IO-SAM}
2. IAM possesses Innocuity {IO-IAM}
3. SAM and IAM are independent {Ind-SAM-IAM}
With bindings
- Innocuity: definition in the OP description {1}
- Independent: to be defined

This snippet shows an argument written in FAN [8]. Note that tags assign unique names to premise and bindings using curly brackets (e.g. {IO-SAM}). Underlined words indicate that the word has specific meaning in this context and is defined within the binding section. As the example shows, the reasoning can be as short as 'conjunction' as long as it is clear that the premises are a sufficient basis for believing the conclusion. In the conjunction case, the reasoning can be also omitted for reducing space.

We connect the textual and the graphical depictions of the argument using the tag attached to each proposition. We use a simple graphical notation as shown in Figure 3. This notation shows the connection between the conclusions. But it intentionally does not provide enough details to suggest the argument can be evaluated using the graphic alone.

C. *Criticality Levels*

We will now move away from the OPs for a bit to explain the origin and purpose of criticality levels and discuss how they are currently handled throughout the development process.

Criticality levels exist to enable a balance between the rigor of required assurance and the cost of obtaining it to differ depending on the anticipated effect on overall system safety. That is, greater assurance is required for those components of the system whose undesired behavior may contribute to severe consequences than for those components whose misbehavior cannot contribute to such consequences.

Current certification practices handle the required assurance throughout the development process through levels described in different guidance documents. The most relevant documents include ones for the safety assessment process (ARP4761, soon to be ARP4761A), system development (ARP4754A, soon to be ARP4754B), for electronic hardware development (DO-254) and software development (DO-178B/C).

Everything starts with the system safety assessment, which identifies possible functional failure conditions and classifies them according to severity (AC 25.1309-1A):

- *Catastrophic*: Failure conditions which would result in multiple fatalities, usually with the loss of the aircraft.
- *Hazardous*: Failure conditions which would reduce the capability of the aircraft or ability of the flight crew to cope with adverse conditions resulting in a large reduction in safety margins.
- *Major*: Failure conditions which would reduce the capability of the aircraft or ability of the flight crew to cope with adverse conditions resulting in a reduction in safety margins.
- *Minor*: Failure conditions which would not significantly reduce the capability of the aircraft or ability of the flight crew to cope with adverse conditions but could result in a slight reduction in safety margins.

Based on the severity, functions get assigned DALs that correspond to the level of rigor that will be required in the development assurance. Severity levels are mapped to DAL levels from A to E, where DAL A is assigned to functions whose misbehavior can cause catastrophic failures in the system. DALs translate into a set of validation and verification processes that aims to reduce errors in the hardware and software development process.

Taking a closer look into software development, DO-178C handles criticality by requiring a set of objectives that differ for each level and by recommending architecture mitigations (e.g., partitioning, multi-version dissimilar software and safety monitoring). Figure 1 shows required specification (arrow going down) and validation/verification (arrow going up) objectives related to the product life cycle. Every set of objectives contains all objective of the lower level (A≥B≥C≥D). Continuous lines correspond to verification done with independence, meaning that the person that reviews an artifact is not the same that created the artifact. The diagram shows that all levels need to test the system against the high-level requirements. Furthermore, with the sole exception of one tier (between source code and object code), all blocks are connected with at least

three lines, meaning the specification and validation/verification is required for levels A, B, and C. That leads us to conclude that an important difference between A, B, and C is the independence in meeting the verification objectives. Another difference not shown in the diagram is the testing coverage metric, which is different for each level (Modified Condition/Decision Coverage for A, Decision Coverage for B, and Statement Coverage for C).

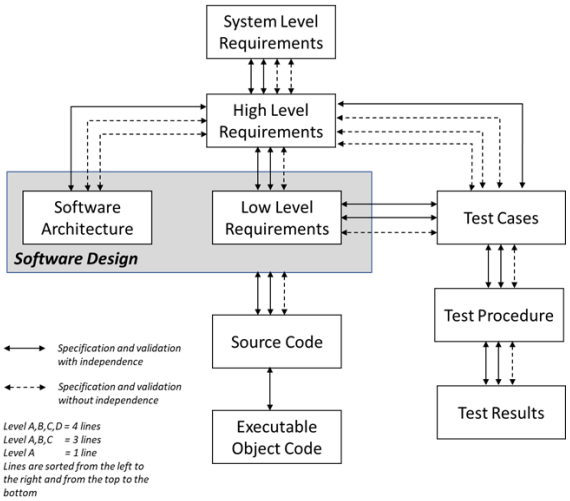

Figure 1. Comparison between DO-178C objectives for DAL A-D grouped based on life cycle products

These objectives were created as a part of DO-178B by a committee that consisted of an international, diverse collection of individuals: toolmakers, academics, certifying authorities, engineers, and managers. These individuals were considered to be experts in their fields. The committee first created a set of objectives to apply to Level A. Afterward, the committee (via a consensus process) voted to remove specific objectives and independence requirements for lower levels. Consensus was required only on whether to remove something, not on the reasons for doing so. Each voting committee member was free to use their own 'gut feel' based on their own experiences. Neither compelling arguments nor empirical data was required. Nevertheless, the set of selected objectives has proven itself to be remarkably successful through service experience over the last three decades and is accepted by approval authorities and applicants.

III. Industrial Case Study

The purpose of the case study is to show how an OPRA for the same industrial system changes to satisfy different criticality levels. The chosen system is the Auxiliary Power Unit (APU) system. It provides the aircraft with electrical and pneumatic power primarily for the main engine's starter motor, heat and air conditioning, supplemental electric power, and potentially in some situations flight power. This system is usually classified as Level C, except when the APU is used for supplemental aircraft propulsion power driving it to Level A. We consciously choose to assign Level D as the criticality level for this paper because 1) the assurance required at Level D is significantly different from the closest levels (C and E) and 2) We believe that certification authorities and applicants will start using OPs for applications with lower criticality levels.

We used two teams in our case study. The applicant team consisted of one Designated Engineering Representative (DER) with 20 years of experience in aerospace certification and one researcher with three years of experience in argumentation. The evaluation team consisted of four people from approval agencies (NASA and FAA) and one argumentation expert from another aerospace company. The applicant team played the applicant role creating the OPRAs while the evaluation team assessed the OPRAs and provided feedback.

The presented OPRA seeks software and system level approval of the APU Control System (APU-CS). The APU-CS is a software-based electronics system that controls APU power generation efficiency, flight deck status reporting, and support for maintenance functions. Functions typically include controlling starter, fuel and bleed systems; reading engine temperatures and pressures; exchanging status information with other aircraft systems; performing Failure Detection and Accommodation (FDA); and transmitting Built-In-Test (BIT).

APU-CS is implemented by (1) hardware components, which are specified in the HW Specification, and (2) software components, which are implemented either by using a Model Based Design (MBD) process or by manual coding (see *Figure 2*). Manually implemented components might represent less than 20% of the entire software and usually correspond to the processor and hardware interface software. AADL models drive the MBD process by specifying high-level functionalities and architecture. Based on AADL models, developers manually create Simulink models that fully implement the functionality. A MathWorks code generator—for which RTCA DO-330 tool qualification credit is not claimed—translates the Simulink models into source code. In this paper, we refer to the AADL models as Specification models (SM) and to the Simulink Models as Design Models (DM). Strong partitioning of this hardware interface software from the "algorithmically based application software" provides for a clean separation between Model-Based components and manual coding. This separation allows the application software to leverage simulation analysis

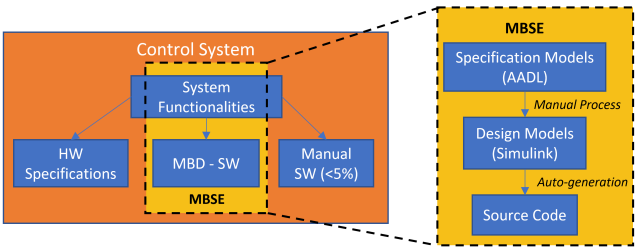

Figure 2. Architecture of APU-CS and development process for the model-based components

and test. We combine testing in the simulation and target platform to verify the software and system level functionality.

Although the APU-CS development process would not seem novel to many MBD practitioners, current standards do not allow use of this process as-is. The main reasons are: 1) DO-331, the model-based supplement to DO-178C, requires "textual requirements above the model" and 2) simulation cannot be used

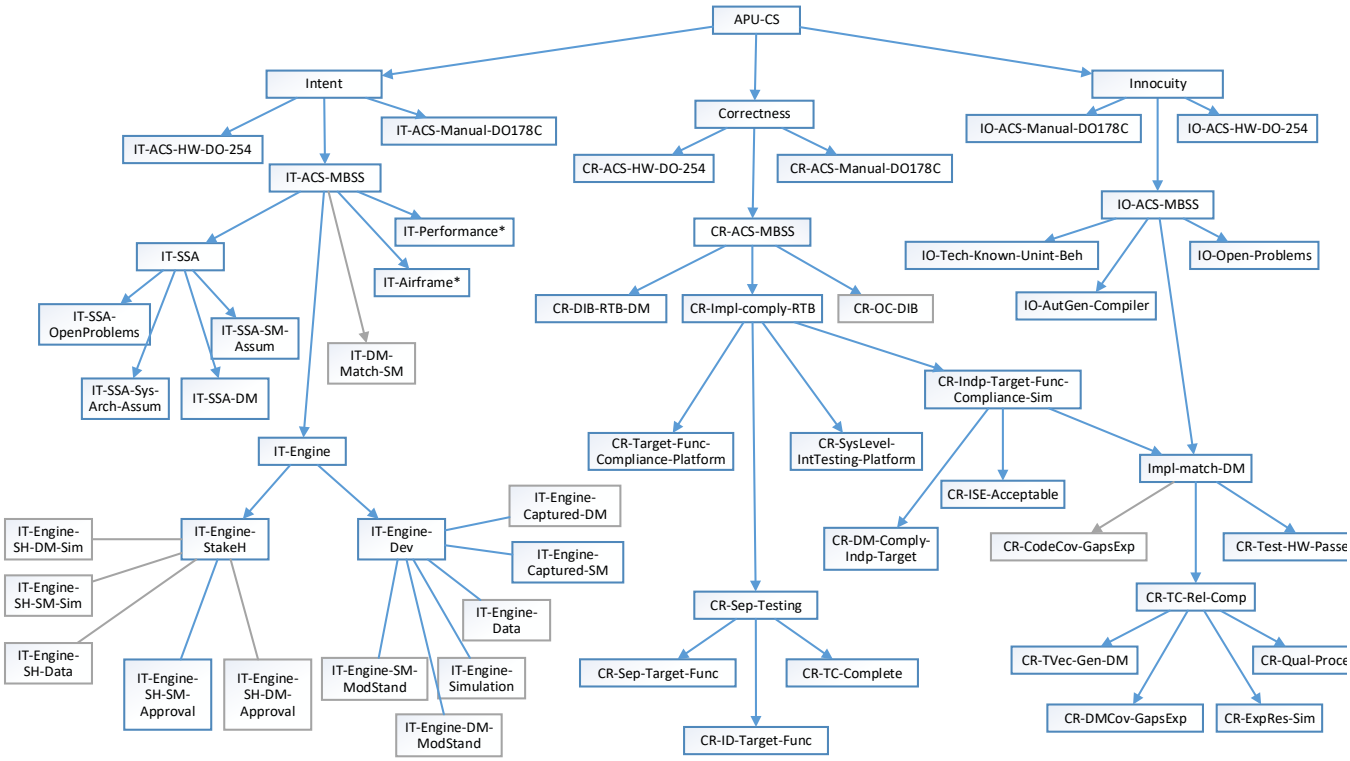

Figure 3. Graphical representation of the OPRA for system and software approval of the APU-CS for Level A (all the nodes) and Level D (only blue nodes)

for showing system correctness. Therefore, applicants are required to add extra steps to seek classical certification approval. That makes the development process suboptimal. Now imagine the frustration of developers that want to create novel safe and optimal development processes for the aerospace industry.

IV. OPRAs FOR APU-CS

This section presents an overview of the OPRAs for Level A and D. Thus, the reader can gain some understanding of the proposed MoC and the difference in the argument between the levels. The Level A OPRA is described in detail in [9]. The following summarizes the proposed MoC for both levels.

- Intent is captured and validated according to both the Stakeholder's perspective and Development Team's perspectives.
- Requirement-driven V&V activities ensure continuity from Stakeholder intent to Implementation and are applied at different abstraction levels of the system.
- Simulation is used in V&V activities for target-independent functionalities in lieu of testing on the target system. For this to be valid, we provided additional V&V activities that ensure:
  - Design models are equivalent to the implementation using automatic test generation
  - Target-related and target-independent functionality can be tested in isolation
  - The simulation environment is representative for target-independent testing
- The system is developed using an MBSE process that enforces the usage of unambiguous models to capture intent and drive design integrity. With this process, source code becomes an intermediate development artifact and therefore it does not require direct verification.
- Test cases for RTB (Requirements for Test Basis) and automatic test generation are evaluated to ensure an appropriated coverage metric.
- Artifacts (argument included) such as plans, models and requirements are controlled in alignment with the current CC1 category.

Due to space limitation, we only show the graphical representation of the OPRA's structure (Figure 3). Every node corresponds to a tagged conclusion, and its textual preposition is in the annex. Leaf nodes are conclusions that are supported by one or multiple artifacts. For example, the conclusion *IT-Engine-Captured-SM* states that developers agree that desired behavior of the Engine is captured in the Specification Models. This conclusion is supported by the result of a review, the procedure defining the review process, guidelines for performing the review, plans and quality assessment of the review, and if the review was validated with independence or not.

Note that the first level of the argument integrates classical standards within demonstration of possession of the OPs. *APU-CS-OPs* states that the APU-CS possesses the OPs if the

hardware complies with DO-254, the hand-written code complies with DO-178C and the ACS-MBSS possesses the OPs. ACS-MBSS corresponds to the system level aspects of APU-CS and the software exclusively developed with MBSE. The scope of that conclusion is defined by DeB and DIB. For Level A, the DIB is defined by Specification Models and Design Models. For Level D, the DIB is defined only by the Specification Models.

The stakeholders of the APU-CS are the Engine team, Airframe team, Performance team and System Safety team. Thus, the DeB is defined by Engine Interface Control Document (ICD), Airframe ICD, Performance ICD and ACS-MBSS System Safety Analysis (SSA). ICDs capture the needs of each stakeholder on a high-level. Thus, the first level of Intent (*ACS-MBSS-OPs*) captures the DIB of the ACS-MBSS correctly and completely by correctly and completely capturing the needs of all stakeholders (*IT-SSA, IT-Engine, IT-Airframe, IT-Performance*) and ensuring that the specification and design models match (IT-DM-Match-SM). The last one is only required for Level A. To ensure that the DIB captures the SSA, the applicant team proposes to validate models (*IT-SSA-SM-Assum, IT-SSA-DM)*, open problems (*IT-SSA-OpenProblems*) and system architecture (*IT-SSA-Arch-Assum*) against the SSA. Design models are only validated for Level A.

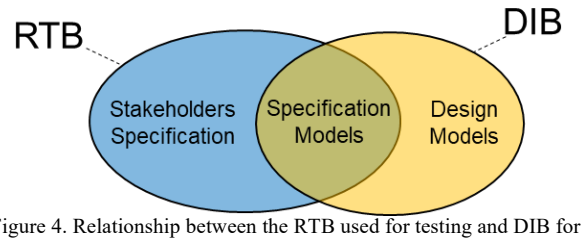

Figure 4. Relationship between the RTB used for testing and DIB for system specification for Level A

The argument for the Engine, Airframe, and the Performance teams is the same. Therefore, for brevity within this paper, the graphic shows only the argument for the Engine stakeholder. The applicant team proposed that the DIB can be said to be correct and complete if the stakeholder together with the developer approve them. This is applicable in this case study because the stakeholders know only the desired behavior at a high-level of their specific part. Thus, the developers can better judge that the entire behavior is captured in a high and low level. For Level A, the stakeholder and developer validate models, simulation results, and data from flight and ground test against the ICD. In contrast for Level D, the stakeholder and developer only validate the specification models.

Correctness of the ACS-MBSS-OPs is shown by testing the implementation (*CR-Impl-comply-RTB*) against an adequate RTB (*CR-DIB-RTB-DB*). The RTB provides two levels of functional testing abstraction: stakeholder needs (e.g., Engine ICD) and specification level. The RTB is required since the DIB for Level A contains the design models (Figure 4). Thus, testing implementation against Design Models will only verify the generator instead of the behavior. In addition, the implementation needs to be validated against some technical details defined only in the ICDs (e.g., ICDs might provide detailed ARINC labels definitions for discrete and analog words that may then only show up in the design models). Therefore, we use RTB for both levels.

For testing the implementation against the RTB, the applicant team divides the functionality of ACS-MBSS into two sets: 1) target-independent set, which is tested using simulation (*CR-Ind-Target-Func- Compliance -Sim*); 2) target dependent set (*CR-Target-Func-Compliance-Platform*), which is tested in the hardware platform. Using simulation for implementation testing is only sufficient if the simulation environment is acceptable (*CR-ISE-Acceptable*), and the design model behavior is equivalent to that of the implementation (*Impl-match-DM).* Thus, applicants can leverage simulation for system-level testing instead of testing every functionality in the target platform, avoiding a potential increase of cost and schedule.

The applicant team proposes to qualify the translation of a specific model by verifying relevant outputs of a given model. Thus, the implementation is equivalent to the design models (*Impl-match-DM*) if the implementation has the same results as the design models (*CR-Test-HW-Passed*) for a complete set of test cases (*CR-TC-Rel-Comp*). Completeness and relevancy of the test cases is achieved by automatically generating test cases from the design models (*CR-TC-Gen-DM*) to completely cover the model structure (*CR-ModCov-GapExp*). To provide confidence in the result, the test generation process is qualified. Note that for Level A, the applicant team adds structural coverage of the code (*CR-CodeCov-GapsExp*) to increase the confidence in the completeness of the test cases.

Innocuity focuses on identifying behavior of the implementation that is not defined by refinement from the DIB and ensuring that this behavior does not cause harm. For the ACS-MBSS, code generator, compiler optimization (*IO-AutoGenCompiler*), open problems (*IT-SSA-Open-Problems*), mismatches between the implementation and the design model (*Impl-match-DM*) are all possible causes of undefined implementation. Thus, the innocuity argument ensures that additional implementation based on all these potential causes will not undermine the safety of the system. Note that Innocuity uses the same premise as Intent's *IT-SSA-OpenProblems* and Correctness's *Impl-match-DM*. In the case of Correctness, *Impl-match-DM* allows applicants to use simulation for testing activities, and in the case of Innocuity, the premise reduces the search of behaviors outside of the DIB since the implementation matches the design model, which is part of the DIB.

There are other differences between the Level A OPRA and the other levels that cannot be observed on the graphical representation, such as, independence requirements of verification/validation activities and supporting artifacts. Independence information is captured on the leaf argument as mentioned earlier. An example for the second aspect is the *IT-SSA*. For Level D, the amount of required BIT, monitors and architectural requirements are likely less because fewer safety considerations apply. Therefore, although the argument and the type of supporting artifacts are the same for all levels, the content of the artifacts may be different, meaning the conclusion or the confidence in the truth of the conclusion may change.

V. Trying a Classical Leveling Approach

After analyzing how current standards handle criticality (Section II.A), we attempted to apply similar leveling techniques to the creation of the OPRA for the APU-CS. The applicant team

followed a similar approach to DO-178C and DO-331. Thus, the applicant team:

1. Created a Level A OPRA (see Figure 3) considering the feedback from the evaluation team
2. Adapted the OPRA for Level D by:
   a. Reviewing every atomic argument and removing premises that were not required for Level D
   b. Removing the Design Model from the DIB
   c. Removing premises related to Design Models
   d. Removing independence Validation Requirements in the artifacts
   e. Removing coverage metrics

Although this process makes superficial sense, when you look at the detailed execution for the specific OPRA, the process removes premises that are important for ensuring the assurance of the proposed MoC. One example is the premises *Impl-Match-DM*. This premise is related to the design model, and therefore, would be removed for Level D if we follow the previous process. The removal of this premise degrades the relevance of the simulation result since we cannot guarantee that the behavior in the simulation corresponds to the behavior in the implementation. For similar reasons, the applicant team also maintains coverage metrics at the model level but removes them at the code level for Level D.

We believe our attempt to apply a traditional approach to leveling taught the following lessons:

The number of premises is not a good indicator of criticality: Reducing the redundant premises can be used to target lower criticality. For example, premises for the validation of the system behavior through model validation (*IT-Engine-Captured-SM*) and simulation validation (*IT-Engine-Simulation*). However, lessening assurance this way must be done thoughtfully and carefully. In particular for dependent premises, such as, the relevance of the simulation environment (*CR-ISE-Acceptable*) and the compliance of the design models shown in the simulation (*CR-DM-Comply-Indp-Target*). In this case, the results that show compliance are only relevant *if* the simulation environment accurately models the target platform.

Reviewing atomic arguments for a given level of criticality is not possible: The required level of assurance is built up through the entire argument. Therefore, assignment of a criticality level to an atomic argument is not possible since it is the context of the atomic argument that provides the assurance sufficiency.

Reducing relevant activities over low-level requirements is not necessarily appropriate: Our design models are analogous to low-level requirements in DO-178C and DO-331, which are not required at Level D. In the case study, we show that this reduction is not directly applicable for the APU-CS since simulation over design models is used to support system verification. Although reducing activities related to low-level requirements is a good way to address lower levels, applicants need to analyze the applicability in their specific OPRA.

The use of OPRAs already optimize and customize the certification process: Current standards provide general guidelines that can be applied for different systems/development processes. However, these guidelines are potentially suboptimal for any specific system/development process. In contrast, an OPRA can specifically target a system, technology, development process or company culture. Therefore, we believe that applicants will propose MoC that are optimal for their development process. That can make it harder to remove premises for lower criticality levels.

Level A bias is a problem: Creating an OPRA for Level A first leads to unconscious bias, thereby, making it hard for the applicant to identify assurance reduction opportunities.

Generically handling criticality is not possible for OPs: Generalization of current standards allows us to use the same standard for different systems and to have a general leveling approach. Since applicants can provide their own customized MoC using OPs, it is not clear we can create a general leveling approach for all possible OPRAs. However, similar argumentation could be reused with precedence for common OPRAs. For example, a OPRA for an engine control system using the same MBSE-based development approach as the APU-CS can use the same APU-CS OPRA.

An applicant is not a committee: Current standards summarize the consensus of a diverse group of experts. In contrast, each OPRA is created by one applicant and approved by the certification authority. On one hand, having only two interested parties makes the approach more agile and optimal for a specific system. On the other hand, having a set of experts that come to a consensus makes standards more trustworthy. OPs are meant to be used for alternative methods or novel technologies. Thus, the applicants are the only ones that know all the details and weaknesses of the methods or technologies. Therefore, there is a need for processes and methods that justify confidence in the sufficiency for the given criticality level of the applicant's proposal.

VI. Leveling an OPRA

The flexibility provided by the OPs allows applicants to propose novel means of compliance. However, the same flexibility makes it theoretically and practically hard to generalize and evaluate the compliance regarding an assigned criticality level. Therefore, we propose a set of self-assessment techniques that aim to help applicants create a strong argument for an assigned criticality level. The self-assessment nature of the techniques is a reaction to the fact that applicants are the ones who know all the details of the proposed development approach. This stands in contrast to certification standards, where the development approaches and their possible risks and mitigations are already known by the certification authorities. This section presents how an OPRA can be adapted to comply to different criticality levels and introduces the self-assessment techniques.

A. Modulation of an OPRA for a proposed criticality

1) Proposed argument: Applicants can propose one means of compliance for a high criticality level and a different one for a low criticality level. For example, an applicant can implement a system using hand-written code from low level requirements for lower criticality levels, and using a MBSE

PRODUCT ARTIFACTS	INPUT ASSURANCE PREMISES	PRODUCT ARTIFACTS	INPUT ASSURANCE PREMISES
SM-SLR/HLR	Stakeholder approved that the engine-related aspects of Specification Models captured the Engine ICD [IT-Engine-SH-Model-Approval]	Source Code	Open problems do not undermine the safety of the system as determined by the SSA [IT-SSA-OpenProblems]
	Engine ICD is correctly captured and completely traceable to the Specification Models [IT-Engine-Captured]	EOC	Open problems do not undermine the safety of the system as determined by the SSA [IT-SSA-OpenProblems]
	System and software architecture satisfy assumptions made by **SSA** [IT-SSA-Sys-Arch-Assum]		Test cases trace and comply with the complete RTB [CR-TC-Complete]
	Failure monitors and BIT in the Specification Models satisfy assumptions made by **SSA** [IT-SSA-SP-Assum]		The implementation complies with the target-related RTB [CR-Target-Func-Compliance-Platform]
	Open problems do not undermine the safety of the system as determined by the **SSA** [IT-SSA-OpenProblems], 26. [IO-Open-Problems]		System operation complies with system level functionality and robustness behaviors [CR-SysLevel-IntTesting-Platform]
	Foreseeable operating conditions are captured in the DIB [CR-OC-DIB]		Analysis of the test cases results run on the target processor shows they match the expected results with accepted tolerance [CR-Test-HW-Passed]
	Target-related functions are identified and tied to the RTB [CR-ID-Target-Func]		
DM-LLR	Open problems do not undermine the safety of the system as determined by the SSA [IT-SSA-OpenProblems], 26. [IO-Open-Problems]		Automatically generated Test Case inputs from Design Models [CR-TC-Gen-DM]
	Test cases trace and comply with the complete RTB [CR-TC-Complete]		Test Cases inputs cover 100% of the model's structure using MC/DC or gaps explained [CR-ModCov-GapsExp]
	System and software architecture ensures that target-independent functions are separated from target-related functions [CR-Sep-Target-Func]		Expected results from Test Cases inputs come from running Design Models in a simulation environment [CR-ExpRes-Sim]
	System operation complies with system level functionality and robustness behaviors [CR-SysLevel-IntTesting-Platform]		Process used to generate the expected results from simulation is qualified [CR-Qual-Process]
	Design models comply with target-independent RTB [CR-DM-Comply-Indp-Target]		Behaviors added by code generator and compiler (e.g., optimization that collapse logic) do not undermine the safety of the system [IO-AutGen-Compiler]
			Functionalities or technologies that are known to cause unintended behaviors are not used in ACS-MBSS [IO-Tech-Known-Unint-Beh]

Figure 5. Result of Product Lifecycle Analysis

approach with qualified code generation for higher criticality levels. Varying the MoC necessarily changes the proposed argument. For example, though this work maintained similar approaches for all criticality levels to facilitate comparison, an applicant team may remove design models from the DIB, thereby leading to changes in the structure of the argument.

2) Supporting artifacts: Artifacts support the leaf premises of an argument. Where argument ends and the supporting artifacts begin is up to the writer. We intentionally stopped at artifacts that contain information relevant to the reasoning that cannot be optimally presented in an argument form. That is, we cited, rather than argued through, results from studies, checklists, plans, and procedures. That writing choice reduced the complexity of the argument, thereby facilitating its creation, evaluation, and configuration control. Thus, supporting artifacts can also be used to address the assigned criticality level. For example, a checklist used for validating requirement compliance can be larger for Level A than for Level D. In the OPRA for the APU-CS, the argument for the Safety Group stakeholder (IT-SSA) is the same for all levels. However, the supporting artifacts vary in content since the assumptions and requirements on the system architecture, monitors, etc. that come from the SSA are likely more numerous.

B. Self-assessment analysis methods

These methods provide a different perspective of the argument and the proposed MoC that help applicants to better judge the sufficiency of the argument against the assigned criticality level. Furthermore, the result of these analysis methods can help evaluators to guide their assessment. In the case study, the applicant performed this type of analysis after the OPRA was finalized and the results of the analysis were provided to the evaluator team together with the OPRA.

1) Product Life Cycle Perspective: aims to force the applicant to think how many V&V levels are addressed over each life cycle product..

Process: List all products of the approach and order the premises of the OPRA according to the product related to the premise (see Figure 5).

Analysis: Evaluate the sufficiency of the V&V layers according to the assigned criticality level.

Based on our experience, certification designees find this view very useful to understand the supporting artifacts. Looking at the table for APU-CS, you can notice that there are not many premises related to the source code. Even for Level D, that could raise a red flag. However, in this specific OPRA, not having premises related to source code is intended, because the applicant team proposes the source code as an intermediate product due to the code generation. This is similar to the usage of compiled code in classical approaches. Note that there are multiple premises related to the design models even though DO-178C does not consider them for Level D. These premises provide more confidence in the simulation used for system level testing but also provide more confidence in the source code.

2) Technology Risk Assessment: aims to force the applicant to identify all risks of the proposed technology and validate if the argument addresses them..

Process: Ask domain experts of all the technologies used in the proposed approach (e.g., model-based design) about possible risks of the technologies without looking at the argument.

Analysis: Map risks to premises used to mitigate them and explain when risk has not been mitigated. Evaluate if the technology risk mitigation is sufficient for the assigned level.

The risks of using Level D OPRA are listed on the bottom of Figure 6. Risk 1 addresses errors during the requirement elicitation process, leading to incompletely and/or incorrectly capturing the needs of the stakeholders. For the APU-CE, system/high-level requirements are captured manually using the specification models from ICDs and the SSA. This risk is mitigated in Intent and Innocuity through validation activities marked in the solid red line. Note that Risk 4 corresponds to deficiencies in correctness. However, there sub-risks are related to the specifics of the development process used. These risks are

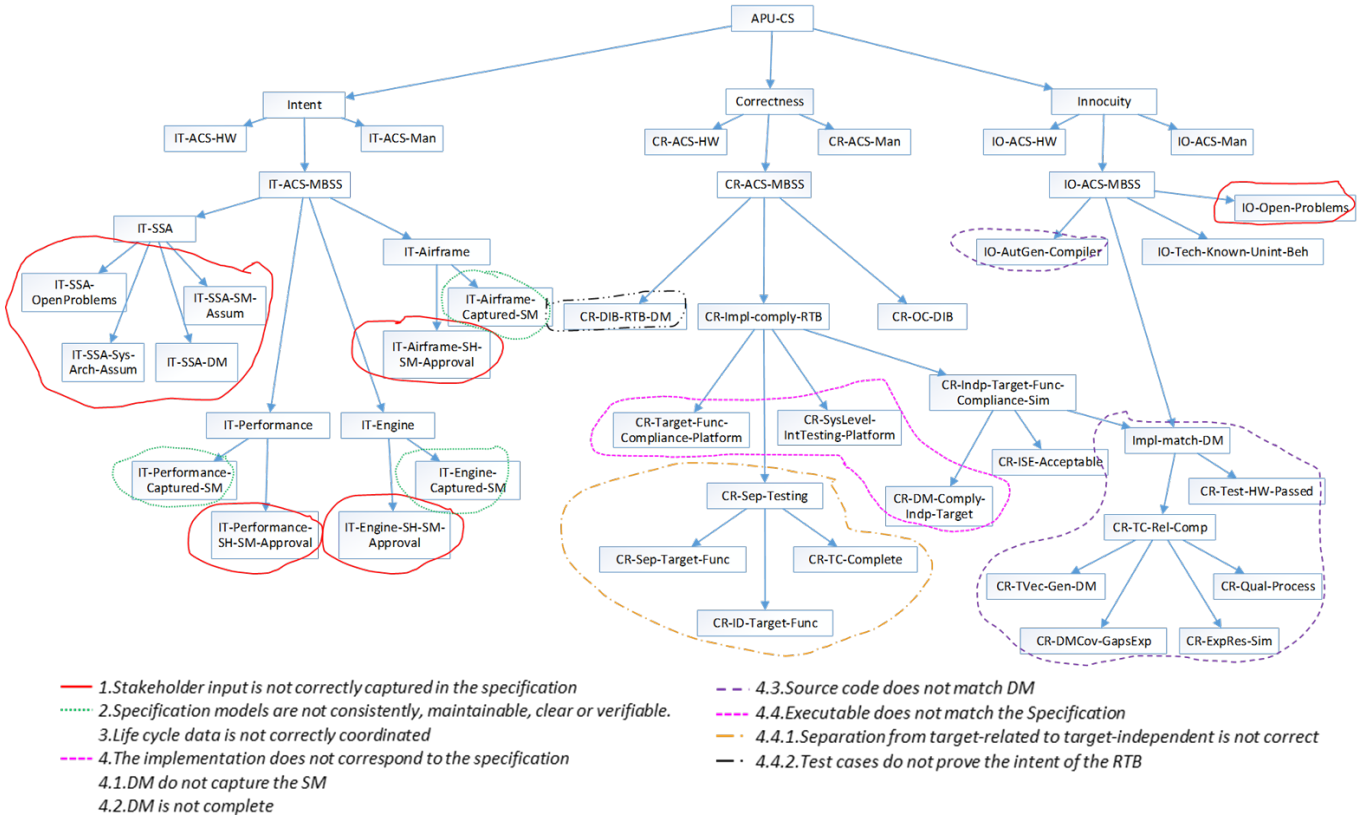

Figure 6. Risks of the technologies used for the development of APU-CS and their mitigation mapping to the OPRA

defined according to design models' refinement (4.1, 4.2), code generation (4.3), the usage of simulation (4.4.1) and testing coverage (4.4.2). For Level D, the applicant team did not address Risks 4.1 and 4.2. However, these risks are addressed for higher criticality levels.

3) Comparison with Level A: aims to force the applicant to reevaluate whether the premises required to achieve DAL A are required for the assigned level..

Process: After creating the OPRA for the assigned level, extend the OPRA to comply with DAL A using the same development process.

Analysis: Compare both OPRAs using the Product Life Cycle view (Including artifacts description and independence) and explain why the presumed additions are not required for the assigned level.

In contrast to classical leveling, we suggest that the applicant starts with the OPRA for the assigned level (e.g., Level D) to avoid bias (see section II.A). This analysis helps applicants to determine if something is missing and evaluators to better understand the proposed method. This analysis is especially helpful for Level D since it is not required that the applicant provide a lot of details about the process, and therefore, evaluators might not be able to evaluate possible impactful aspects of the approach. A part of the result of this analysis for the APU-CS is shown in Figure 7. As expected, there are more premises related to the design model than for Level D. However, there are premises that appear in both. Furthermore, you can see that all of the activities are performed with independence.

PRODUCT ARTIFACTS		INPUT ASSURANCE PREMISES
DM-LLR	I	Analysis of the simulation of the Design Models shows that DIB captures functional content defined in the Engine ICD [IT-Engine-SH-Sim]
	I	Stakeholder approved that the engine-related aspects of Design Models captured the Engine ICD [IT-Engine-SH-Model-Approval]
	I	Engine ICD is correctly captured and completely traceable to Design Models [IT-Engine-Captured]
	I	Analysis of the data from running Design Models in an integrated simulation shows consistency with the Engine ICD [IT-Engine-Simulation]
	I	Design models and Specification Models follows Modeling standard [IT-Engine-ModStand]
	I	Behaviors added (e.g., behaviors that lead to floating point exceptions) by the Design Models (not in the Specification Models) do not undermine the safety of the system as determined by the SSA [IT-SSA-DM]
	I	Open problems do not undermine the safety of the system as determined by the SSA [IT-SSA-OpenProblems]
	I	Foreseeable operating conditions are captured in the DIB [CR-OC-DIB]
	I	Test cases trace and comply with the complete RTB [CR-TC-Complete]
	I	System and software architecture ensures that target-independent functions are separated from target-related functions [CR-Sep-Target-Func]
	I	System operation complies with system level functionality and robustness behaviors [CR-SysLevel-IntTesting-Platform]
	I	Design models comply with target-independent RTB [CR-DM-Comply-Indp-Target]
	I	ISE is an acceptable environment for testing target-independent RTB [CR-ISE-Acceptable]

Figure 7. Comparison focused on premises related to design models between OPRAs for Level A and Level D

VII. Conclusion

Our case study on industrial examples indicates that OPRA leveling should not be approached in the same manner as traditional leveling practices. We attribute this difference to the high flexibility and customization of OPs, which makes it difficult for applicants to offer the same level of assurance as a diverse consensus-based committee. The case study demonstrated that modifications to the argument structure and supporting artifacts enable applicants to tailor an OPRA based on assigned criticality levels. To reduce argument complexity,

we propose to support leaf premises by one or multiple artifacts that can contain a significant amount of information. Consequently, these supporting artifacts can have a substantial impact on the criticality level. Further research is necessary to gain a better understanding of the interaction between an OPRA and its supporting artifacts. Recognizing that the applicant possesses comprehensive knowledge of the proposed MoC, we recommend the adoption of self-assessment methods to strengthen the argument according to the criticality level. While our industrial case study sheds light on the challenges of leveling OPRAs, conducting additional diverse industrial case studies seems necessary to provide valuable insights into the intricacies of the process.

ACKNOWLEDGMENT

Kimberly Wasson, George Romanski, Mallory Graydon, and Michael Vukas have participated in this project throughout, contributing insightful ideas, corrective comments, and fun facts, for which we thank them! They deserve credit along with us for some of the good ideas described in this paper. But not for any errors or bad ideas that may be herein.

REFERENCES

[1] C. M. Holloway, "Understanding the Overarching Properties," NASA/TM-2019-220292. July 2019. [https://ntrs.nasa.gov/citations/20190029284]

[2] C. M. Holloway and K. Wasson. "An Introduction to Constructing and Assessing Overarching Properties Related Arguments (OPRAs): Version 1.0" NASA White Paper, 2022. [https://ntrs.nasa.gov/citations/20210025425]

[3] M. A. Aiello, C. Comar, and J. F. Ruiz, "An Assurance Case based on Overarching Properties for QGen: a TQL1 Code Generator" 10th European Congress: Embedded Real Time Systems, 2020.

[4] J. Chelini, J. Camus, C. Comar, D. Brown, A.-P. Porte, M. de Almeida, and H. Delseny, "Avionics Certification: Back to Fundamentals with Overarching Properties," in HAL archives-ouvertes, 2018.

[5] H. Forsberg and A. Schwierz, "Emerging cots-based computing platforms in avionics need a new assurance concept," in 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), pp. 1–8, IEEE, 2019.

[6] M. Graydon, and J. Cronin (2021). "Retrospectively documenting satisfaction of the Overarching Properties: An exploratory prototype" NASA/TM-20210014715).

[7] M. Durling, H. Herencia-Zapana, B. Meng, M. Meiners, J. Hochwarth, N. Visser, and V. Valapil. "Certification Considerations for Adaptive Stress Testing of Airborne Software". In 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC) (pp. 1-9). IEEE.

[8] Z. Daw, S. Beecher, C. M. Holloway and M. Graydon. Overarching Properties as means of compliance: An industrial case study. In 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC) (pp. 1-10). IEEE.

[9] Z. Daw, and S. Beecher, "Assuring safety in a flexible aerospace certification—Lessons learned on applying OPs at the system level—". In 2023 IEEE International Systems Conference (SysCon) (pp. 1-8). IEEE

[10] C. M. Holloway, "The Friendly Argument Notation (FAN)," ASA/TM-2020-5002931. June 2020. [https://ntrs.nasa.gov/citations/20205002931]

[11] J. Rushby, X. Xu, M. Rangarajan, and T. L. Weaver. "Understanding and evaluating assurance cases." NASA Contractor Report NASA/CR-2015-218802, NASA Langley Research Center, July 2015

[12] R. Bloomfield, P. Bishop, C. Jones, and P. Froome, "ASCAD—Adelard safety case development manual," Adelard, vol. 5, 1998.

[13] European Aviation Safety Agency, "AMC 20-115C software considerations for certification of airborne systems and equipment," European Aviation Safety Agency, December 2013.

ANNEX

Definitions used in the premises can be found in [9].

IT-Engine-SH-Sim: Analysis of the simulation of the Specification Models shows that DIB captures functional content defined in the Engine ICD
IT-Engine-SH-Model-Approval: Stakeholder approved that the engine-related aspects of Specification Models captured the Engine ICD
IT-Engine-Captured: Engine ICD is correctly captured and completely traceable to the Specification Models
IT-Engine-ModStand: Specification Models follows modeling standard
IT-SSA-Sys-Arch-Assum: System and software architecture satisfy assumptions made by SSA
IT-SSA-SP-Assum: Failure monitors and BIT in the Specification Models satisfy assumptions made by SSA
IT-SSA-OpenProblems: Open problems do not undermine the safety of the system as determined by the SSA
CR-OC-DIB: Foreseeable operating conditions are captured in the DIB
CR-ID-Target-Func: Target-related functions are identified and tied to the RTB
IT-Engine-SH-Sim: Analysis of the simulation of the Design Models shows that DIB captures functional content defined in the Engine ICD
IT-Engine-SH-Model-Approval: Stakeholder approved that the engine-related aspects of Design Models captured the Engine ICD
IT-Engine-Captured: Engine ICD is correctly captured and completely traceable to Design Models
IT-Engine-Simulation: Analysis of the data from running Design Models in an integrated simulation shows consistency with the Engine ICD
IT-Engine-ModStand: Design models and Specification Models follow Modeling standard
IT-SSA-DM: Behaviors added (e.g., behaviors that lead to floating point exceptions) by the Design Models (not in the Specification Models) do not undermine the safety of the system as determined by the SSA
IT-SSA-OpenProblems: Open problems do not undermine the safety of the system as determined by the SSA
CR-OC-DIB: Foreseeable operating conditions are captured in the DIB
CR-TC-Complete: Test cases trace and comply with the complete RTB
CR-Sep-Target-Func: System and software architecture ensures that target-independent functions are separated from target-related functions
CR-SysLevel-IntTesting-Platform: System operation complies with system level functionality and robustness behaviors
CR-DM-Comply-Indp-Target: Design models comply with target-independent RTB
CR-ISE-Acceptable: ISE is an acceptable environment for testing target-independent RTB
IT-SSA-OpenProblems: Open problems do not undermine the safety of the system as determined by the SSA
IT-Engine-SH-Data: Analysis of the ground/flight-testing data shows that the implementation is consistent with the Engine ICD
IT-Engine-Data: Analysis of the ground/flight-testing data shows that the implementation is consistent with the Engine ICD
IT-SSA-OpenProblems: Open problems do not undermine the safety of the system as determined by the SSA
CR-TC-Complete: Test cases trace and comply with the complete RTB
CR-Target-Func-Compliance-Platform: The implementation complies with the target-related RTB
CR-SysLevel-IntTesting-Platform: System operation complies with system level functionality and robustness behaviors
CR-Test-HW-Passed: Analysis of the test cases results run on the target processor shows they match the expected results with accepted tolerance
CR-CodeCov-GapsExp: Test cases run on target processor with instrumented code show suficient coverage or gaps justified
CR-TC-Gen-DM: Automatically generated Test Case inputs from Design Models
CR-ModCov-GapsExp: Test Cases inputs cover 100% of the models structure using MC/DC or gaps explained
CR-ExpRes-Sim: Expected results from Test Cases inputs come from running Design Models in a simulation environment
CR-Qual-Process: Process used to generate the expected results from simulation is qualified
IO-AutGen-Compiler: Behaviors added by code generator and compiler (e.g., optimization that collapse logic) do not undermine the safety of the system
IO-Tech-Known-Unint-Beh: Functionalities or technologies that are knownto cause unintended behaviors are not used in ACS-MBSS