Overarching Properties as means of compliance: An industrial case study

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

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

Michael Holloway, Mallory Graydon
Safety Critical Avionics Systems Branch
NASA Langley Research Center
Hampton, Virginia
c.michael.holloway@nasa.gov
m.s.graydon@nasa.gov

***Abstract*—The Overarching Properties (OPs) have been created by an international working group and are being evaluated by the National Aeronautics and Space Administration (NASA), the Federal Aviation Administration (FAA), industry, and other certifying agencies in an effort to streamline certification processes. Their intent is to facilitate the use of alternative approaches and to allow flexibility to combine the system, software, and complex hardware certification. The hope is that the FAA may eventually establish an Advisory Circular that offers the OPs as a Means of Compliance (MoC) for software approval (and eventually systems and hardware) by showing the product possesses the three OPs: *Intent* (specification of the intended behavior), *Correctness* (implementation of the intended behavior) and *Innocuity* (safety of unintended behavior). In the certification community, there is still a concern about the practicability of using such high level properties in certification. This paper aims to address that concern by showing possession of the OPs in an industrial case study using assurance arguments. The two main contributions of this paper are: a certification process based on OPs as Means of Compliance, and a certification argument for an on-board physical model of an UAV (Unmanned Aerial Vehicle), as industrial example. We propose a hybrid approach for the certification process that combines OPs with existing certification standards. Thus, OPs can be used for parts of a system that uses technologies that are not supported by current standards or for which existing standards require additional effort without commensurate additional safety assurance.**

***Index Terms*—overarching properties, certification, arguments, assurance cases**

I. Introduction

Aerospace systems are seeing a strong trend towards ‘electronification’. This move towards increasingly critical and complex functionality and safety behaviors captured in aerospace electronics is driving consideration of new certification methods. Systems complexity is challenging certifying agencies, approving organizations, academia and industry to search for new and innovative technologies to provide effective assurance of safety-based systems. Certification applicants, like Unmanned Aerial System (UAS) makers, have complained (rightfully so) to certifying agencies that today’s standards do not account for the complexity of newer technologies. In some cases the technology or methods used do not provide sufficient safety assurance to support them. Furthermore, the fast development and adoption of new technologies has outpaced the development of new standards. Therefore, the FAA has proposed consideration of streamlining the aircraft certification and approval processes. In relation to this streamlining effort, the Overarching Properties (OPs) concept has been developed by an international Overarching Properties Working Group (OPWG) with NASA and certifying agencies including FAA and EASA support. The intent of this group is to develop the OPs concept to a point where the certification agency could recognize it as a Means of Compliance (MoC). OPs are a set of three high level properties that define a sufficient set of properties for making approval decisions. The high-level nature of the properties offers applicants great flexibility when certifying a system since these properties are not specific to a domain or a type of system. This flexibility could make certification more agile and adaptable to evolving technologies. In contrast to existing standards, arguments that combine hardware, software, and system characteristics can be leveraged using OPs.

Certifying agencies have expressed some concern on the practical use of these high-level properties. We identify two main challenges in adopting OPs as a means of compliance: the heterogeneity and the verifiability of the argument. Due to the flexibility of the OPs, applicants can submit an argument that shows that the systems hold the OPs using any format. This can make applications difficult to evaluate and can significantly slow down the approval process. To present the argument, applicants can for instance create a white paper with a textual description of the argument. With so many worldwide aerospace development organizations, copious use of this open free-form method without standardization would likely cause confusion and difficulty to interpret the argument. This in turn would cause inconsistencies in the evaluated results as assessed by certification authorities. We use assurance arguments to present OPs arguments in a more systematic, consistent, and complete manner. Assurance arguments are also used in the literature [1]–[4] to create a well-structured argument for OPs. An assurance argument is defined as an explicit argument that a system or service is acceptable for its intended use [5]. To emphasize the centrality of arguments, we use the term assurance argument henceforth instead of assurance case. The structure provided by the assurance arguments can facilitate the evaluation of the OPs, addressing the verifiability problem. The heterogeneity of the argument (i.e. every applicant can come with their own argument) was considered to potentially overwhelm certification authorities. This led to discussions of argumentation patterns and limiting use of OPs for specific portions of systems which are not supported by current recognized standards or which require additional effort using these standards without the additional safety assurance. Assurance argument patterns are generic arguments that a type of system or service is acceptable for a given use under specified restrictions. An approved library of argumentation patterns can increase the confidence in the arguments and speed up the approval process of a specific system.

Although there have been some initial efforts in different companies to apply OPs in the certification process, showing possession of OPs is still a process in its infancy. There have been some research efforts in this area. Collins Aerospace is currently carrying out an OPs project in conjunction with Adacore and their QGEN product as a TQL1 qualification with FAA oversight [1]. There are some published works that use assurance arguments to show possession of OPs. [3] discusses how to define evaluation criteria using assurance arguments and demonstrate them for a micro UAV. [4] also uses assurance arguments to provide airworthiness arguments for Commercial off-the-shelf (COTS) components. [2] retrospectively created documentation showing that SAFEGUARD, an assured safety net technology for UAS, holds the OPs using assurance arguments. This paper also uses assurance arguments to show possession of the OPs. In contrast, we propose a hybrid approach that combines OPs with existing certification standards (Section III). In addition, this paper presents a certification process that supports the use of OPs as a Means of Compliance, and detailed assurance arguments (Section V) based on an industrial example (Section IV) that aims to clarify how to bridge the gap between the high-level properties and system evidence. Section VI presents lessons learned from applying OPs.

II. Background

A. Overarching Properties

In 2015, the FAA started the effort to streamline certification processes. This includes a move towards process audits based on company reputation, facilitating the use of alternative approaches, and allowing integrated certification of systems, software and complex hardware. The OPs themselves were first presented to the public in late 2016, and further developed and refined over the last several years through work by NASA and the OPWG. They were developed from distillation of existing standards and research-based contemplation for what is needed for complete assurance. It has informally been shown that these properties capture the foundational concepts upon which the existing standards are built. Beyond the properties themselves, NASA and OPWG have developed supporting definitions and proposed a structure for property-based argumentation. The expectation is that the FAA may eventually establish an Advisory Circular that provides a Means of Compliance (MoC) for software approval (and eventually systems and hardware) by showing the product possesses the OPs. This will be “a means but not the only means” of approval and offer an alternative to AC 20-115D that recognizes DO-178C and its associated document suite. The following is the definition of the OPs [5]:

- ***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.

The OPs also define: a) **Requisites** for showing possession of the OPs, in which the applicant must demonstrate that the system and its safety assessment exist; b) **Assumptions** that the applicant has about the system in relation to its behavior and safety; and c) **Constraints** on how to show possession of the OPs by showing correctness and completeness of the process (e.g. safety assessment must address all the implementation).

The OPs concept may be extended beyond a single discipline to include systems, software and hardware. For example, one might jointly certify a board with an FPGA (Field-Programmable Gate Array), its firmware, and its integrating software.

B. Argument specification using assurance cases

In most OP-related literature, assurance arguments are used to show that a system possesses the OPs. Assurance arguments provide a consistent and reusable structure that facilitates the communication and evaluation of an argument. Assurance arguments are used to present an argument that a system is safe, secure, reliable, etc. within a given context. There are many notations one could use to represent an assurance argument. Some of these are textual [6], others graphical, such as Goal Structuring Notation (GSN), Claims Argument and Evidence (CAE) notation, and the new Structured Assurance argument Metamodel (SACM) notation [6]–[8]. The presented methodology is not aligned with any specific notation or tool. Currently, in this project, we use the Friendly Argument Notation (FAN) [9] to specify the assurance arguments. FAN is a textual notation that is based on a small number of terms chosen to promote clarity, understanding and communication of arguments and facilitate the use for assurance arguments by ordinary engineers. FAN defines the following concepts:

- **Argument:** An attempt to convince others to believe a conclusion through reasoning and one or more premises.
- **Believe:** Accept as true.
- **Premise:** A statement you think your audience believes.
- **Reasoning:** States why you think the premises should cause your audience to believe your conclusion.

- **Binding:** An association between a term used in an argument and the real-world information to which that term refers.
- **Defeater:** Statement that may cause your audience to not believe your conclusion.
- **Conclusion:** The statement you want your audience to believe.

Figure 1 shows an assurance argument specified in FAN [9]. Note that every premise, reasoning, conclusion, or binding can be tagged with a unique name, which is depicted using curly brackets (e.g. 1 in Figure 1). Underlined words in the proposition 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.

Believing
Subsystems SAM and IAM both possess Innocuity {1}
Is justified by applying
the principle of conjunction {2}
To these premises

1) SAM possesses Innocuity {3}
2) IAM possesses Innocuity {4}
3) SAM and IAM are independent {5}

with these definitions

- Innocuity: definition in the OP description ¡https://hdl.handle.net/2060/20190029284¿ {6}
- independent: to be defined {7}

Fig. 1: FAN example from [5]

III. Hybrid Certification Approach

Existing standards provide a good safety and security guideline. However, there are systems or parts of systems that use technologies that are not supported by current standards or which require additional effort when using these standards without the additional safety assurance. Aviation regulation for engine control software 14CFR33.28(g) states that “The applicant must design, implement, and verify all associated software to minimize the existence of errors by using a method, approved by the FAA, consistent with the criticality of the performed functions”. Since regulations do not state a specific method to use for compliance, we propose a combination of existing standards (RTCA DO-178C / EUROCAE ED-12C) and OPs, called the hybrid approach, as a means of compliance. The motivation of the hybrid approach is to leverage existing standards where they are effective, and to use OPs for parts of the software that are not as well supported by the standards. Note that this approach applies equally well for systems and hardware approvals or for multidisciplinary approvals. This section presents the means of compliance for the hybrid approach and the corresponding certification liaison process as applicable for this industrial aviation software example.

A. *Certification Argument*

From an argumentation point of view, existing standards provide an implicit argument for certification that has been evaluated and accepted by a group of experts. Thus, certification authorities only evaluate whether the executed activities and generated artifacts satisfy the objectives of the existing standard. Using OPs, certification authorities need to also evaluate the argument, which we called a certification argument. The certification argument shows that a system or a component holds the OPs. The certification argument is presented in the form of an assurance argument that specifies this argument in a structured manner and includes both (a) binding documents, which provide background information relevant to the argument, and (b) a list of artifacts that support the leaf premises of the assurance argument.

During its development, the certification argument is a living document that may change during the planning and development process of a product. As development proceeds, the applicants may add to or revise both the argument and its supporting evidence. Figure 2 shows the proposed life cycle.

Argument planning process: Applicants construct a certification argument of the proposed premises that satisfy the conclusions that the product holds the OPs. This process produces the Planned Certification Argument (PCA), which is a snapshot of the certification argument presented with the Plan for Software Aspect of Certification (PSAC).

PCA approval process: The certification authority reviews the PCA and provides rebuttals that have to be addressed by the applicant. This interaction between applicants and certification authorities continues until an agreement is reached. The expectation is that the approved PCA is the baseline certification argument and approved with the PSAC.

Argument completion process: Applicants update and refine the approved certification argument incrementally as needed based on project execution adjustments. This process produces the Argument Accomplishment Summary (AAS), which is the project’s final snapshot of the certification argument presented with the Software Accomplishment Summary (SAS). Throughout project development, changes to the certification argument must be tracked (CC1). In addition, relevant changes in the certification argument shall be coordinated with the certification authority.

AAS approval process: The certification authority reviews the AAS and provides rebuttals that the applicant must address. This interaction between the applicant and the certification authority continues until an agreement is reached. The expectation is that the approved AAS is approved with the SAS.

B. *Certification Liaison*

The certification liaison establishes a process to ensure communication and understanding between the applicant and the certification authority throughout the argument life cycle.

C. *Means of Compliance and Planning*

The PSAC as defined in RTCA DO-178C / EUROCAE ED-12C is commonly used to define the means of compliance

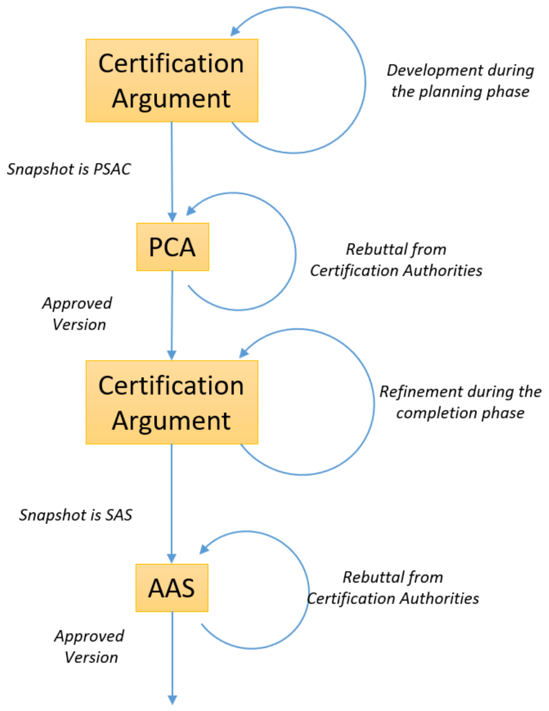
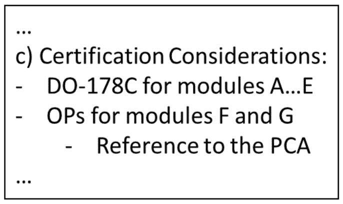

Certification Argument
Development during the planning phase
Snapshot is PSAC
PCA
Rebuttal from Certification Authorities
Approved Version
Certification Argument
Refinement during the completion phase
Snapshot is SAS
AAS
Rebuttal from Certification Authorities
Approved Version

Fig. 2: Certification Argument Life Cycle

for software projects. In order to support the proposed hybrid approach, we extend the Certification Considerations section 11.1.



…
c) Certification Considerations:
- DO-178C for modules A…E
- OPs for modules F and G
    - Reference to the PCA
…

Fig. 3: Example of an extended PSAC to support a hybrid certification

The Certification Considerations section summarizes the proposed means of compliance to satisfy the regulations. As in all software approvals for certification, including this hybrid certification, applicants shall provide the failure condition categories of the components, and the Means of Compliance such as DO-178C or OPs (see Example in Figure 3). For components using OPs, applicants shall include or reference the Planned Certification Argument (PCA), which provides the argumentation for certification basis. The PCA consists of a hierarchical assurance argument that shows that the target component holds the OPs. The PCA provides sufficient information for the evaluator to understand the means of compliance. Details regarding the depth of the hierarchical assurance argument, meaning amount of argumentation levels, shall be determined in agreement with the certification authority. Argumentation levels are created when a premise is further argued by putting it as the conclusion for a lower-level component of the argument. For the PCA, the assurance arguments shall provide a complete argument without linking the actual evidence. That is achieved by providing enough information about the content of the artifacts that support the argument. In most projects, the PSAC and supporting plans would reference procedures, work instructions, review forms and other supportive artifacts that provide additional detail to the PCA evaluator.

D. *Compliance Substantiation*

At project completion, the applicant provides the evidence that satisfies the PSAC which is summarized in the SAS. For components using OPs, the applicants shall provide the AAS, which consists of the complete argument with access to the supporting artifacts, identified with references or links. Throughout project execution, the certification authority must be made aware of any structural or significant changes in the argument.

E. *Quality Assurance*

For Quality Assurance (QA), we believe this function should be performed in a similar manner as to what is done for objectives in the current standards. We want QA to review and ensure execution of all project plans, and adherence to the means of compliance (including OPs assurance arguments) as identified in the PSAC and PCA. The Quality Assurance process results should be summarized in the same Conformity Report as identified for use with existing standards. The QA team should take a reviewer's role in the creation of the certification arguments. The QA process for the certification argument aims to provide confidence that:

1) The planned argument defined by the assurance arguments satisfies OPs.
2) The final assurance arguments are compliant with the planned assurance arguments.
3) The supporting artifacts comply with the requirements of their artifact types and support the claims they were asserted to support.

IV. Case Study: Robot dynamic model for UAV with a model-based collision avoidance

This paper discusses an industrial aviation example that addresses software for a robotic dynamic on-board model (ON-RDM) of an UAV. The overall system is a UAV with a model-based collision avoidance system. A collision is avoided by exploiting the dynamic behavior of the robot and the measured relative distances between objects in the environment for automatically determining control inputs to safely steer the UAV away from obstacles. Model-based controls can be found in other applications, in which a model helps to estimate parameters that are used for the control module [10], [11] or for diagnostic purposes. This model could be considered a surrogate for other software, systems or hardware complex logic that has no specific requirements for the internal intermediate values. We propose using OPs in a hybrid manner where argumentation is proposed as the means of compliance for the ON-RDM only. Existing standards are proposed where they are effective for the remaining aspects of the system. We believe this hybrid approach to be a typical use of the OPs, at least in the near future.

On-board models can be algorithmically complex with physics-based polynomial computations. These models generally have no internal or intermediate parameters used within the functional or safety aspects of the system. Current standards require detailed traceability and for decomposed requirements to be thoroughly tested. In cases like models and AI/ML type logic, we often do not care about the micro-behaviors except for safe execution (exceptions, etc.). This on-board model macro-behavior focus can be shared with AI/ML logic where specific detailed responses are not required, but rather safely bounded responses are considered acceptable. Flight and failure scenarios are learned and must be safe but perfectly optimized responses are not required or even possible.

Although FAN provides a binding section, we propose to additionally provide a binding document that provides an overview of the system that highlights technologies and methods that are relevant to the assurance argument evaluation. This overview provides more technical details than is typically provided in a PSAC. The following subsection illustrates a proposal for content of the binding document. It is important to mention that this is only a part of the actual binding document, and by no means complete.

A. *ON-RDM – Background*

The ON-RDM is a real-time robot dynamics model, which estimates the robot dynamics in run-time based on sensed values taking into account physical deterioration, faults, weather, etc. Here the UAV control can be designed to maximize performance without excessive conservatism on the robot dynamics. There is a second, off-board robot dynamics model (OFF-RDM) that can serve as an input to the robot development team. While the ON-RDM is specific to one robot model line and requires validation, the OFF-RDM is a widely used generic model that has been extensively analyzed and validated. The OFF-RDM is widely used to simulate and analyze robot dynamics and is already accepted by stakeholders due its high accuracy. If it could be shown that the ON-RDM is not only suitable for real-time use but a sufficiently close approximation of the OFF-RDM, stakeholders would likely accept it as well.

The accuracy of the OFF-RDM is related to the individual accuracy of the model components of the vehicle. Therefore, the model of the components needs to get calibrated using data from the real UAV data. The validation timeline of an OFF-RDM is shown in Figure 4. OFF-RDM$_{V_0}$ is instantiated based on stakeholder requirements (e.g. navigation vector). This serves as specification for the production of the UAV. Data is extracted from testing of the UAVs. This data is used to calibrate OFF-RDM$_{V_0}$, resulting in OFF-RDM$_{V_1}$. Based on OFF-RDM$_{V_1}$, the ON-RDM is developed.

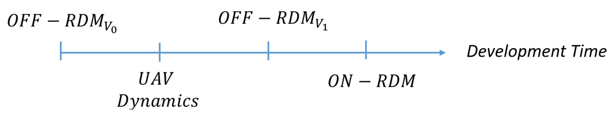


Fig. 4: Development timeline for the ON-RDM

Although the OFF-RDM model can accurately estimate robot dynamics, the model can require a great amount of memory and processing resources and can need non-deterministic amounts of execution time to provide an answer. These aspects make it difficult to use and approve them in a run-time environment. The UAV control algorithm requires that the model provides outputs for navigational estimation at a given periodic rate. So non-deterministic algorithms are generally deprecated for safety critical aerospace applications considering embedded processors have limited computational resources. The ON-RDM guarantees a timing determinism that safely supports the navigational estimation by sacrificing a prescribed level of accuracy in the parameters. The definition of accuracy, in this context, is the difference in the estimation between the OFF-RDM and ON-RDM, which is due to the difference in the resolution of the physics based models and computation models. The required accuracy of the estimation of a parameter depends on the usage of that parameter in the UAV control algorithm. Depending on the usage of the model's output, as input to the control within the most sensitive environmental condition, a conservative accuracy margin for every output is established. However, this margin can be relaxed on a case-by-case basis based on an acceptability analysis while still providing a safe margin, which guarantees that the robot dynamics control behaves safely in the computation of the vehicle's navigational vectors.

V. Methodology for creating an OPs argument

For showing OPs, assurance arguments are used to present an argument in a structured manner showing that the system or a component of a system holds the three OPs, *Intent*, *Correctness*, and *Innocuity*. These assurance arguments consist of different hierarchical levels of argumentation and can be quite large. For the sake of space, this section is focused on the parts of the assurance argument for the ON-RDM that are unique to the OPs: a) the first level of the argument that creates the bridge between OPs and the assurance arguments, and, b) a leaf assurance argument that connects the premises with existing artifacts. Moreover, this section presents the argument structure of the PCA and the AAS for the ON-RDM.

We use a combination of textual and graphical notation to represent the assurance arguments. The textual notation is the source of the argument and the graphical notation can be extracted from it. Based on our experience, we believe that textual representation allows the writer to focus on the argument without distraction of the structure, in contrast to the graphical representation. We are using FAN as a textual representation. However, we recognize that the big picture of the argument is key in the evaluation of the OPs. Therefore, we propose a simple graphical notation that provides an architectural overview of the argument structure, but it is not meant to give the reader enough details to fully evaluate the argument. We connect the textual and the graphical depictions of the argument using a tag attached to each proposition. Our convention is to use short tag names that signal the meaning of the propositions they identify. We separate words with dashes and begin tags with prefixes that identify the argument the propositions appear in: IT for *Intent*, CR for *Correctness*, and IO for *Innocuity*. Although the presented assurance arguments provide additional information in the bindings section, a binding document summarizing the technical approach, justifications, and definitions is also presented to the certification authorities. For the case study, we create one assurance argument for every property, which creates 3 primary argument branches with one each for the 3 OPs.

A. *Intent*

To show possession of the *Intent* property, the assurance argument needs to argue that the applicant is accurately capturing the intent of the stakeholders. The premises for this argument, which will be agreed with stakeholders, will identify applicable artifacts and verification activities such as reviews and analyses. In keeping with current standards, intent is often captured in system-level requirements, and further detailed with software high and low-level requirements. As explained earlier in section IV, software low-level requirements would not meaningfully capture what the ON-RDM must do (or not do). Accordingly, we specify and verify the macro-behavior of the model, the behavior of the robot dynamics, using a highly accurate physical-based robot dynamic model as ground truth.

Figure 5 presents our argument that the ON-RDM possesses the *Intent* property. In the first argumentation level, we rely on the formal definition of the property for the justification, as you can see in the bindings. We customize the definition to the target system by redefining what is DIB, DB and correct and complete. In order to define the DB, it is important to identify first who is the stakeholder of the ON-RDM. In this case, it is the UAV Control Module, which specifies the accuracy needed in the outputs of the estimation model.

The definition of completeness states that the intended behavior is only complete if the robot dynamics (macro-behavior), accuracy, robustness and timing requirements are specified. In contrast, the intended behavior is correct if there is a functional equivalence between the robot dynamics and the ON-RDM.

Now that we know what needs to be achieved (conclusion) and how we are planning to achieve it (reasoning), we can start with the premises that support the argument. Premises 2, 3, and 4 address accuracy, robustness, memory and timing requirements, respectively. Note that definition of robustness is based on a standard definition. However, the definition is also customized to the ON-RDM by stating possible abnormal conditions specific to on-board models. Accuracy is defined as the difference between the on-board model (ON-RDM) and the off-board model (OFF-RDM), which is a highly accurate

Believing
ON-RDM **holds** Intent {Intent}
Is justified by applying
ON-RDM defined intended behavior is correct and complete with respect to the DB
To these premises

1) The OFF-RDM is a highly accurate model representation of the robot dynamics {IT-OFF-RDM-baseline}
2) The ON-RDM is accurate in relation to robot dynamics as needed by the control {IT-accuracy}
3) The OFF-RDM is used as a surrogate robot dynamics model for accuracy specification for each ON-RDM output parameter {IT-accuracy-spec}
4) ON-RDM robustness requirements are correct and complete for off-nominal input values and numerical instability {IT-robustness}
5) ON-RDM requirements correctly address all memory and timing constraints {IT-mem-time}

with these definitions

- ON-RDM: ON-board Robot Dynamic Model
- OFF-RDM: OFF-board Robot Dynamic Model
- Intent: The Defined Intended Behavior (DIB) is correct and complete with respect to the Desired Behavior (DB) (Definition from OPs). Customization to this project:
  DIB = Requirements of ON-RDM
  DB = Model of the robot dynamics at a level of accuracy as needed by the control system
- Correct and complete:
  - The complete detailed description of the behavior of the ON-RDM correctly captures the robot dynamics plus accuracy, robustness, memory and timing requirements defined by the control system {ON-RDM-behavior}
  - *Correctness* within the context of the intent of ON-RDM means functional equivalency to the robot dynamics
- Highly accurate: aims to qualify the accuracy of the ON-RDM and the OFF-RDM in relation to the robot dynamics
- Accuracy: represents the difference between the ON-RDM against the robot dynamics, which is defined by the difference of the ON-RDM output parameters against the OFF-RDM output parameters under the same inputs due to premise [IT-OFF-RDM-baseline]
- Robustness: corresponds to "The extent to which software can continue to operate correctly despite abnormal inputs and conditions" (Definition from RTCA DO-178C). Additional to abnormal inputs and conditions common to embedded software, ON-RDM is affected by other abnormal conditions, such as, instability due to discrete mathematical abstraction and abrupt changes in the inputs
- Needed by the control: means that the control system sets the flight envelope and the accuracy of each output parameter

Fig. 5: Intent of ON-RDM

representation of the robot dynamics as stated in premise 1. Thus, correctness is fully addressed by premises 1 and 2. All the premises are supported by additional assurance arguments (see in argument structure). In order to evaluate the validity of the argument, evaluators need to ask themselves, assuming that the premises are true, "can the premises fully support the conclusion." If one believes these premises, can one agree with the conclusion? In the case that there is doubt that a premise can be achieved, for example it can be hard to believe that the OFF-RDM is so accurate that it can be used as a reference or ground truth, the evaluator can validate the acceptability of the premise by validating the supporting argument for that premise.

Believing
ON-RDM holds Correctness {Correctness}
Is justified by applying
ON-RDM implementation is correct w.r.t requirements of ON-RDM under the flight envelope
To these premises

1) The ON-RDM implementation is a representation of the robot dynamics at the level of accuracy as needed by the control system {CR-accuracy}
2) ON-RDM implementation is robust for off-nominal input values and numerical instability {CR-robustness}
3) ON-RDM implementation operates within memory and timing constraints {CR-mem-time}

with these definitions

- Correctness: The implementation is correct with respect to its Defined Intended Behavior (DIB), under foreseeable operating conditions. [Definition from OPs]. Customization to this project:
DIB = Requirements of ON-RDM defined in intent
Foreseeable operating conditions = Flight envelope
Requirements of ON-RDM = robot dynamics, accuracy, robustness, memory and timing requirements
- Flight envelope: In aerodynamics, the flight envelope defines operational limits for an aerial platform with respect to maximum speed and load factor given a particular atmospheric density. The flight envelope is the region within which an aircraft can operate safely
- Correct within the context of the correctness of the ON-RDM means functional equivalency to the ON-RDM requirements
- The accuracy of the ON-RDM implementation against the robot dynamics is defined by the accuracy of the ON-RDM implementation against the OFF-RDM assuming premise IT-OFF-RDM-baseline
- Robustness: corresponds to "The extent to which software can continue to operate correctly despite abnormal inputs and conditions" (Definition from RTCA DO-178C). Additional to abnormal inputs and conditions common to embedded software, ON-RDM is affected by other abnormal conditions, such as, instability due to discrete mathematical abstraction and abrupt changes in the inputs

Fig. 6: *Correctness* of ON-RDM

Believing
ON-RDM holds *Innocuity* {Innocuity}
Is justified by applying
Any part of the executable ON-RDM that is not required by the DIB has no unacceptable impact
To these premises

1) Open/retained problems represent non-required behaviors {IO-open-problems}
2) Unexpected behaviors caused by implementation choices, such as, numerical instability, inaccuracy, and exceptions are addressed by the DIB {IO-impl-choises-DIB}
3) Other unexpected behaviors caused by implementation choices that are not addressed in the DIB are uncovered by system regression testing and robustness testing {IO-impl-choises-noDIB}
4) Safety assessment addresses all open/retained problems and unexpected behaviors caused by implementation choices {IO-safety-assess}
5) Open/retained problems and implementation choices have no unacceptable impact as concluded by the safety assessment {IO-safety-impact}
6) Common software functionalities or technologies that might cause unintended behaviors are not used in ON-RDM implementation {IO-NA-common-unint-beh}

with these definitions

- Innocuity: Any part of the implementation that is not required by the Defined Intended Behavior (DIB) has no unacceptable impact. Customization to this project:
Implementation = Executable ON-RDM
DIB = Requirements of ON-RDM captured or defined in intent (robot dynamics, accuracy, robustness, memory and timing requirements). Any additional requirements added during the development process are added to the DIB.
- Open/retained problems: Throughout verification activities when the implementation does not match the requirements a problem report is created. Based on a control board assessment, the problem can remain open for a specific software release.
- Implementation choices: unintended behaviors that can emerge due to approximation of the OFF-RDM (accuracy), and due to additional behaviors related directly to the ON-RDM implementation. ON-RDM Implementation:
  - ON-RDM can be implemented using component-based physics equations, look-up tables, approximations using linear equation systems, physic-based algorithmic approximation or neural networks.
  - In this specific implementation physic-based algorithmic approximation, unintended behaviors are numerical instability due to discretization of the calculation and the constraints of the target platform, such as timing and memory.
- Common software functionalities or technologies: External developed libraries, COTS, multi-tasking and multi-core, Non-deterministic algorithms, and machine learning functions.

Fig. 7: *Innocuity* of ON-RDM

B. Correctness

In this property, the assurance argument needs to specify how the applicant proposes to show that the implementation matches the intent. Figure 6 presents the assurance argument that shows that ON-RDM holds *Correctness*. Similar to Intent, the official definition is customized to the system and to the foreseeable operating conditions, i.e. the target environment of our system, which in this case is the drone's flight envelope. The reasoning states that the implementation of the ON-RDM is correct if there is functional equivalency with the ON-RDM requirements specified in *Intent*. Note that premises for *Correctness* partially mirrors the premises from *Intent*. The only one that does not have a reflection in *Correctness* is premise 1 of Intent since that premise is not related to the implementation of the ON-RDM.

C. Innocuity

In this property, the assurance argument needs to specify that the implementation contains no behavior that undermines safety. We approach *Innocuity* a little differently than the other properties, in which we first assume that there are no behaviors outside of the DIB and try to come up with the defeaters of the argument. Later, the defeaters are transformed into premises. Figure 7 presents the assurance argument that shows that the ON-RDM holds *Innocuity*. The official definition of *Innocuity* is customized in the same manner than the other properties. There are two types of behaviors presented in the implementation of the ON-RDM that are not specified in the DIB, such as, open/retained problems (premise 1) and unexpected behavior caused by implementation choices

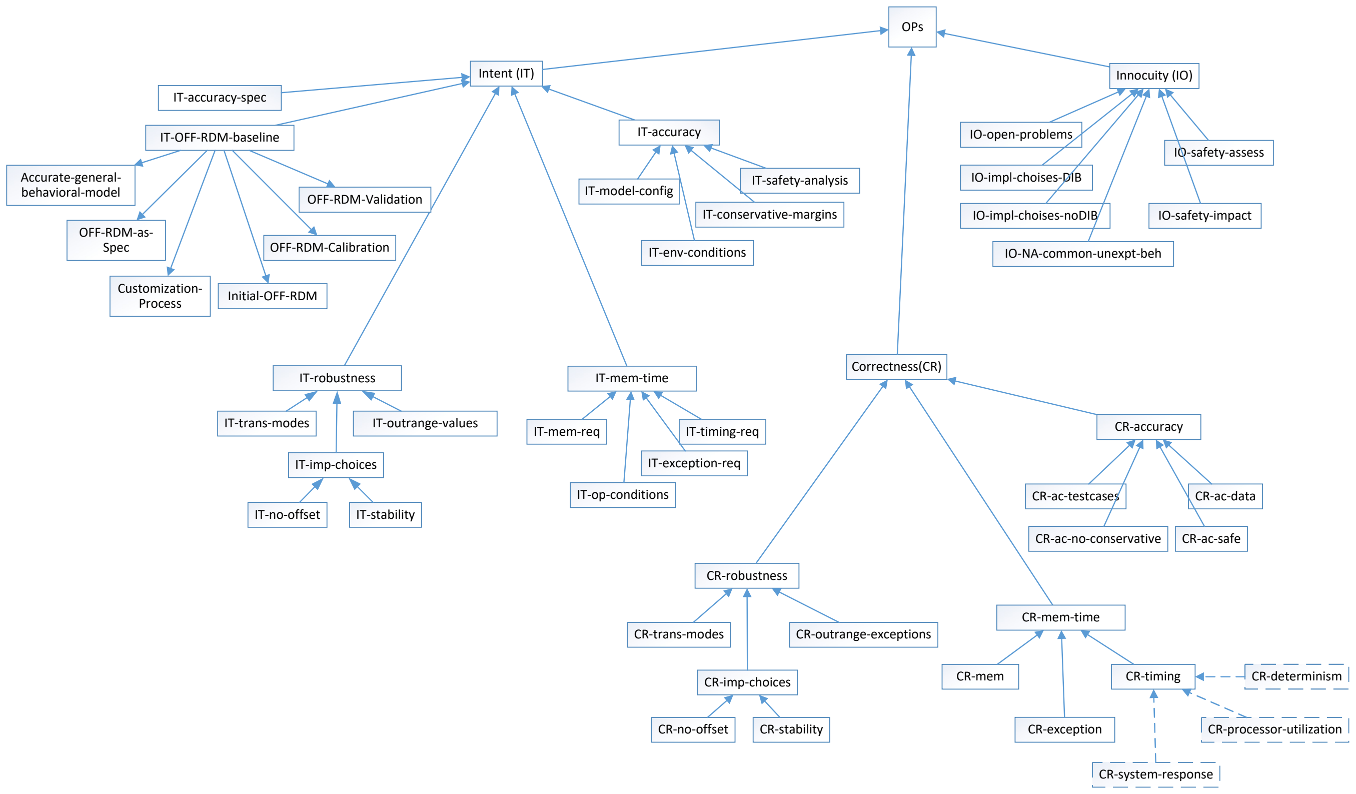


Fig. 8: AAS for the ON-RDM. Dotted argument is added to the PCA during the execution phase.

(premise 2). Premises 4 and 5 provide assurance that these additional behaviors do not impact the safety of the system by performing a safety assessment.

Open/retained problems refers to parts of the implementation that do not completely satisfy their intent (e.g. timing response slower as expected). In the industry, open/retained problems are common and must be carefully analyzed and accepted by the Safety team. ON-RDM is an approximation of the robot dynamics and can be implemented in different ways from a look up table to a neuronal network. Depending which type of implementation is chosen, there are unexpected behaviors caused by that implementation (e.g. timing delays, inaccuracies, non-determinism). In the case of the ON-RDM, a physics-based algorithmic approximation is chosen. This implementation choice can lead to numerical instability due to discretization of the calculation and the timing and memory constraints of the target platform. Although these behaviors are known, premise 2 states that these specific behaviors are addressed in the DIB. This aims to facilitate the understanding and the evaluation of the argument. Additional behaviors caused by the implementation are uncovered by regression and robustness testing.

Once we addressed all behaviors outside of the DIB, we evaluate whether common sources of additional behaviors are applicable to the ON-RDM implementation. Premise 6 groups them and states that they are not applicable. Although this premise can be counter intuitive, it can help evaluators to understand that the applicant had already evaluated and validated those sources of additional behaviors. The argument without this premise is still valid. However, this premise can help the evaluation process and the re-usability in other contexts.

D. *Argument structure*

The argument structure is presented in Figure 8 and aims to provide the evaluator an architectural overview of the argument. This high-level overview is especially important for showing that OPs are held since there is some flexibility in where to address any specific topic. For example, depending on how the DIB is defined, some behavior specification and validation could move from *Intent* and *Correctness* to *Innocuity*. Or in the case of the ON-RDM, additional behaviors produced by the implementation choice are instead addressed in *Intent* and *Correctness*. We believe that the high-level overview can help evaluators to verify and validate the completeness of the argument.

The argument structure is visually helpful but does not contain enough detail to evaluate the argument. It is intended to be examined together with the assurance arguments presented in FAN. The structure graphically connects a conclusion to the corresponding premises. Note that if a premise is supported by further argument, that premise is also the conclusion of that other argument and is, in turn, supported by other premises. This structure can be automatically generated from the assurance arguments.

OPs connect the conclusion that “ON-RDM hold the OPs”, to the previously presented arguments. As shown in Figure 5, *Intent* is supported by premises related to accuracy (IT-accuracy), robustness (IT-robustness), memory and timing (IT-mem-time), and the ground truth model (IT-OFF-RDM-baseline), and therefore, they are connected in the argument structure. All these premises are supported by another assurance argument. Note that the arguments for *Correctness* mirror the structure of the arguments for *Intent*. We believe that this mirroring facilities the clarity on the proposed verification and validation process.

PCA consists of the argument that is depicted with solid lines. The doted-line argument is added during the completion phase. Thus, the structure overview shows the complete argument presented as AAS. During the planning phase, applicants provide a description of the type of artifacts that support the premise. The actual artifacts are referenced and made available after the completion phase. Note that only the premise of CR-timing is extended in the AAS using an additional assurance argument. This premise/conclusion states that “ON-RDM implementation satisfies timing intent”. After the completion phase, CR-timing is supported by the following premises: the estimation is completed within timing constraints as required by the control (CR-system-response), the implementation does not exceed the allocated processor budget (CR-process-utilization), and code execution is deterministic (CR-determinism). The latest premise is required by the calculation of the timing and processing utilization, which are only accurate for deterministic code. It is expected that some premises of the PCA are refined in the AAS, as well as, some minor changes in the argument structure. However, significant or architectural changes should be discussed with certification authorities.

E. Leaf argument

A leaf argument is the level where additional premise detail is not needed. Leaf arguments provide premises that are believable in light of referenced artifacts, with no additional decomposition needed. Leaf arguments are identified on a premise basis, where a conclusion may have one or more leaf premises and one or more premises requiring additional decomposition for agreement with the conclusion.

Figure 9 shows the leaf argument that states that the implementation accuracy is sufficient for the UAV control. In this case, all the premises are meant to be supported by evidence from specific artifacts. Although the majority of the artifacts that support the assurance argument are created during the completion phase, applicants still need to provide enough information for evaluating the argument, including the planned evidence. Therefore, we propose to provide types of artifacts instead of the artifact content for the PCA. The artifact type describes the scope and nature of how the artifact supports the premise. Information about artifacts is collected in a table to facilitate the evaluation of the existence of the evidence (see Table I).

Premise Tag	Artifact Type	ACC
CR-ac-testcases	Verification Cases and Procedures document	CC1
CR-ac-data	Verification Test Report, Model Test Report	CC2
CR-ac-no-conservative	Model Report	CC2
CR-ac-safe	Model Report	CC2

TABLE I: Description of supporting artifacts, their connection to the assurance argument and their Artifact Control Category (ACC)

Believing
The ON-RDM implementation is a representation of the robot dynamics behavior at the level of accuracy as needed by the control system {CR-accuracy}
Is justified by applying
Estimated outputs from the ON-RDM implementation are accurate if the output is within a defined general conservative accuracy margin or an additional acceptability analysis that ensures that ensures that the UAV control behave safely.
To these premises
1) Test cases and procedures established to encompasses the entire envelope are identified {CR-ac-testcases}
2) Execution data of the ON-RDM outputs are collected {CR-ac-data}
3) Outputs of the ON-RDM implementation outside of the conservative margin are identified {CR-ac-no-conservative}
4) Accuracy of outputs outside of the conservative margin does not compromise the safety of the UAV control {CR-ac-safe}

with these definitions
- Accuracy: Represents the difference between the ON-RDM against the robot dynamics behavior, which is defined by the difference of the ON-RDM output parameters against the OFF-RDM output parameters under the same inputs. This is valid due to premise {IT-OFF-RDM-baseline} is true
- General conservative accuracy margin: defines a conservative accuracy range of every parameter depending on the usage of the model’s output as input to the control logic within the most sensitive environmental condition. This margin ensures the safety of the system but it can be relaxed on a parameter basis
- Acceptability analysis: evaluates whether the accuracy of output, which is outside of the conservative margin, does not compromise the safety of the UAV control

Fig. 9: FAN example from [9]

VI. Lesson Learned

In this section, we want to share some observations from working this project that might be helpful for applicants and certification authorities that want to use assurance arguments for showing possession of OPs.

1) Adopting assurance arguments in the certification process
   - The creation of assurance arguments was mainly performed by one argumentation expert and one experienced certification designee. This team structure helped to create well-formed assurance arguments that contain realistic information relevant to the certification process. We believe that this team structure will

- be required during the adaptation phase of assurance arguments in the certification process.
- Flexibility provided by the OPs may potentially overwhelm certification authorities, leading to discussions of argumentation patterns and limiting use of OPs for specific portions of systems. These may be portions which are not supported by current recognized standards or which require additional effort using these standards, without the additional safety assurance.
- To write effective argumentation with the assurance arguments components, a change in mindset might be required from an activity-based mindset to a proposition-based mindset. The activity-based mindset is commonly utilized in the certification process since current standards use it to satisfy objectives. In contrast, a proposition-based mindset incorporates statements that can either be true or false.

2) Argumentation for OPs
   - There is no official order in which the three properties must be addressed. Based on our experience, we see a lot of value in addressing *Intent* first since it is very useful to understand the intended behavior to know what needs to be verified in *Correctness* and to identify what behaviors are not intended and need to be addressed in *Innocuity*. This project benefited from addressing *Correctness* before *Innocuity* since some verification results in *Correctness* were used for validating some premises in *Innocuity*.
   - The key role of the binding information can be underestimated. Although the other components of the assurance arguments provide the argument structure, the bindings give information that provide credibility in the propositions and to avoid misinterpretation, especially if common words are used (e.g. correct).
3) Although this research focused on the creation instead of the evaluation of assurance arguments for OPs, we identified two different evaluation steps: 1) evaluation of the wellformedness of the argument and 2) the evaluation of the validity of the argument. For the second one, system and technology information is needed. We think that the binding documents are key for evaluating the validity of the argument.
4) Representation of assurance arguments: There are different notations for assurance arguments (see section II). The selection of the notation is not straight forward if there is no prior experience. During this research, we tried a light version of GSN as graphical notation, a tabular notation and FAN as textual notation
   - In the brainstorming phase, the graphic notation got in the way of the idea flow since we were more focused on the pictures. In contrast, both textual notations were very useful to discuss the argument. However, tabular notation (Excel table) was not suitable to add binding information. FAN felt more intuitive for specifying the assurance argument.
   - Although FAN helped us to focus on the argument, it was hard to see the overview of the argument, which has proven to be important when creating arguments for OPs. Therefore, we propose to complement the textual notation with a graphical representation of the argument structure providing a visual architectural view. This graphical representation is a convenience only; it is not part of the argument itself.
   - The level of detail to be presented in the graphical representation is not straight forward. Putting all details in the graphical representation becomes unreadable. Abstracting some information has the risk that readers can take the graphical representation as the complete argument forgetting that some information has been abstracted. Therefore, we proposed to only provide an identifier forcing the reader to review the complete assurance argument. This can be easily automated from FAN.
   - While the project was only a small sample of a much larger system, the argument diagram and the FAN argument can be structured to represent a hierarchical organization. This could decompose a system into sub-components where sub-components have assurance arguments that collectively support the system. Arbitrarily large systems could be supported by the proposed notations.

VII. Conclusion

Assurance methods using Overarching Properties have a clarity and power through the identification of the true key properties that are needed to establish assurance. Currently there are several standards with several dozens of objectives (like DO-178C, DO-254, and ARP4754A) that have integrated some level of technology or development assumptions into the standard's often detailed requirements. OPs provide properties that address 'what' needs to be accomplished as opposed to 'how'. With today's technologies and methods evolving faster than regulation and standards can keep up with, the OP approach is of interest for aerospace safety assurance when used in conjunction with existing standards. The ability to focus on a component or provide an argument across disciplines adds to the power of OP assurance. In order to understand how to use OPs as means of compliance, the three high-level properties have to be grounded to practical properties, objectives and activities for system approval. This paper presents a hybrid certification approach that applies OPs to one component of a UAV, the on-board robot dynamic model, for which existing standards are not effective. We also provide a summary of the certification argument for this component and lessons learned during this research project that, we hope, helps the reader to understand how to use OPs as a means of compliance in a practical manner. An important point of this industrial example is the proposed activities that specifically do not match existing standards. The intent is to enable applicants and certification authorities to accept novel methods for compliance that involve arguments and artifacts different from those implied or required by existing standards.

Future works are focused on moving beyond a hybrid approval of a software component to a more comprehensive systems approval with inclusion of the System Safety Assessment. Additionally, there is interest in cross discipline approval where systems, software and hardware aspects of a component can be approved with an OP argument. Furthermore, there is still research needed on the creation and evaluation of argumentation patterns that can facilitate the recognition of OPs as a means of compliance by gaining consensus across industry, academia, and particularly across certifying authorities.

VIII. Acknowledgment

This research was supported by NASA through grant no. 80NSSC20M0006. We are thankful to George Romanski and Barbara Lingberg from the FAA for their essential contribution during this research project. We would also like to thank Overarching Properties Working Group for their feedback and relevant discussions.

References

[1] 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.

[2] M. Graydon, “Retrospectively Documenting SAFEGUARD’s Possession of the Overarching Properties,” in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks–Supplemental Volume (DSN-S)*, pp. 27–28, IEEE, 2019.

[3] 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.

[4] 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.

[5] C. M. Holloway, “Understanding the Overarching Properties,” *NASA Report*, 2019.

[6] J. Rushby, “The interpretation and evaluation of assurance cases,” *Comp. Science Laboratory, SRI International, Tech. Rep. SRI-CSL-15-01*, 2015.

[7] R. Bloomfield, P. Bishop, C. Jones, and P. Froome, “Ascad—adelard safety case development manual,” *Adelard*, vol. 5, 1998.

[8] OMG, “Structured Assurance Case Metamodel (SACM),” 2015.

[9] C. M. Holloway, “The Friendly Argument Notation (FAN),” 2020.

[10] D. Bareiss, J. R. Bourne, and K. K. Leang, “On-board model-based automatic collision avoidance: application in remotely-piloted unmanned aerial vehicles,” *Autonomous Robots*, vol. 41, no. 7, pp. 1539–1554, 2017.

[11] S. Garg, “Aircraft turbine engine control research at NASA Glenn research center,” *Journal of Aerospace Engineering*, vol. 26, no. 2, pp. 422–438, 2013.