# Applying ACVIP for Verification by Analysis during Airworthiness Qualification

Charles Payne, Adventium Labs, Minneapolis, MN
John Hudak, PE, CMU SEI, Pittsburgh, PA
Bruce Lewis, Adventium Labs, Minneapolis, MN
Dr. John McGregor, CMU SEI, Pittsburgh, PA
Alex Boydston, US Army, Redstone Arsenal

DISCLAIMER: This material is based upon work supported by the U.S. Army Combat Capabilities Development Command Aviation & Missile Center (AvMC) under contract no. W911W6-17-D-0003/ W911W6-20-F-703A with Adventium Labs and FA8702-15-D-0002 with Carnegie Mellon University Software Engineering Institute (CMU SEI). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views or endorsement by the U.S. Army.

## Abstract

In this paper, we demonstrate that the Architecture-Centric Virtual Integration Process (ACVIP) provides value for military aircraft airworthiness qualification. Military aircraft airworthiness criteria describe aviation airworthiness processes and the criteria, standards, and methods of compliance necessary for airworthiness assessment of manned and unmanned military aircraft systems. The U.S. Army Military Airworthiness Certification Criteria (AMACC), for example, includes elements from many existing civilian standards and is used to define airworthiness requirements for existing and new acquisition programs. Software safety of complex systems is assured by compliance with formal development processes and testing of essential elements. The AMACC also allows for verification by analysis to detect defects in the evolving software design. Going forward, the U.S. Department of Defense's (DoD) Digital Engineering Strategy will improve aircraft requirements, design, and development through model-based engineering.  ACVIP provides the foundation needed for effective model-based verification by analysis.

## Executive Summary

This paper highlights the value of model-based virtual integration and analysis for airworthiness qualification. The goal of the paper is to guide use of the Architecture-Centric Virtual Integration Process (ACVIP) by military aircraft airworthiness authorities when qualifying aircraft according to program-specific requirements derived from military aircraft airworthiness criteria. For the purposes of this paper, our exemplar is the U.S. Army Military Airworthiness Certification Criteria (AMACC), but our recommendations may also apply to other organizations, such as the U.S. Navy and U.S. Air Force, that define their own airworthiness criteria.

The AMACC describes the U.S. Army Aviation Airworthiness processes and the criteria, standards, and methods of compliance necessary for airworthiness assessment of U.S. Army manned and unmanned aircraft systems. Software safety of complex systems is assured by compliance with formal development processes and testing of essential elements. The AMACC also allows for verification by analysis to detect defects in the evolving software design. Despite these safeguards, industry-wide experiences reveal that airworthiness qualification costs continue to rise, because with the increased use of software, there is a corresponding increase in unmitigated software integration failures.

To slow this trend, the U.S. Army has embraced the U.S. Department of Defense's (DoD) Digital Engineering Strategy for model-based development and has created the Architecture-Centric Virtual Integration Process (ACVIP) to apply model-based analyses as part of model-based virtual integration. ACVIP analyses expose software integration flaws early in the software engineering process while the cost to fix them is still low. ACVIP leverages the SAE International Aerospace Standard (AS 5506) for the Architecture Analysis & Design Language (AADL) to target the embedded software and hardware such as that used in the U.S. Army Future Vertical Lift (FVL) aircraft.

Model Based Systems Engineering (MBSE) coupled with ACVIP will improve the AMACC's verification by analysis. Model analysis evidence against a semantically rigorous model, along with complementary model analysis evidence from different analysis domains to validate those results, will help the airworthiness authority gain more objective evidence by which to assess the qualification risk in the evolving software design. Applying ACVIP will help Army program managers gain standard methods to generate or obtain that evidence.

To demonstrate that ACVIP improves verification by analysis, we apply ACVIP to a particular AMACC latency requirement. The *ACVIP Modeling and Analysis (M&A) Handbook* instructs how to apply ACVIP latency analysis as the model evolves. Early analysis will confirm whether performance budgets are sufficient for latency requirements, and as the model adds detail, later analysis will verify whether those budgets continue to be sufficient. ACVIP also exposes potential weaknesses in the latency result. During the conduct of the latency analysis, the modeler might assume that resource contention and scheduling constraints do not delay software execution and that safety faults or component interactions do not lead to deadlock. The airworthiness authority cannot validate these assumptions within the latency result itself, and the latency analysis is incomplete unless those assumptions can be validated. To address these concerns, the *ACVIP M&A Handbook* identifies complementary analyses to apply to the model that will detect components that could deadlock, components that could exceed their performance envelope, and components that could fail in a particular state. In addition, the *ACVIP M&A Handbook* identifies analyses that reveal safety and cybersecurity hazards whose mitigations might increase latency. ACVIP analyses support the AMACC's comprehensive system safety program.

ACVIP will also support the military's objectives for airworthiness qualification. U.S. Army Regulation (AR) 70-62, for example, provides the charter for the process of assessing the airworthiness of Army aviation Systems. Per AR70-62 the Commanding General of the U.S. Army Aviation and Missile Command will "ensure that systems safety engineering and management principles, criteria, and techniques are applied to achieve optimally low safety risk." ACVIP will help guide the development of embedded systems to low risk and provide strong analytical support for the verification of their non-functional requirements as well as the correctness of integrated interaction behavior across components. By adopting ACVIP for the AMACC's verification by analysis for embedded systems, the Army will gain a significant capability to understand, verify, and update its embedded systems more rapidly. This capability translates into reduced complexity for the airworthiness authority engineer verifying a model. It also translates into increased coverage of requirements through formal and automatable analysis to complement testing. Given the airworthiness authority's role to assess the impact of change on airworthiness, and to require requalification if necessary, virtual integration will provide a new capability for making this assessment across multiple domains of analyses (e.g., database loading, processor loading, memory loading, timing, etc.) prior to implementing the change.

ACVIP will augment existing processes for airworthiness qualification. For example, the AMACC's Preliminary System Safety Analysis (PSSA) process identifies points where individual components are analyzed and the integrated system is evaluated. Augmenting this process with ACVIP activities will change the effort profile of the process. Defects can be detected early in the process, reducing the effort required later in the process. The *ACVIP Acquisition Handbook* recommends that the government program office produce a template ACVIP Plan that outlines the ACVIP evidence to include at the major milestones and reviews. The contractors in turn will produce ACVIP Management Plans that respond to the ACVIP Plan to provide the details of how their development process will address the issues raised in the ACVIP Plan. Early on, the government's Airworthiness Qualification Plan should incorporate the ACVIP Plan, and the contractor's response, or Airworthiness Qualification Specification, should incorporate the ACVIP Management Plan.

Finally, we describe considerations that will simplify the airworthiness authority's transition to evaluating model analysis evidence for verification by analysis. These considerations include modeling software, clarifying the modeling workflow, obtaining semantic consistency across various modeling languages (e.g., FACE, SysML, AADL, etc.), adapting verification by test to use ACVIP results, and training stakeholders. As a first step, the Army Program Manager (PM) and contractors will want to agree on the set of AMACC requirements that will benefit from ACVIP-generated evidence. Next, the PM and contractors will want to agree on appropriate ACVIP analyses to cover those requirements. Model verification analysis should map to testing of the implemented system. Finally, the PM and contractors will want to agree on the levels of model maturity that align with program review and/or established model maturity milestones. These agreements will support definition of the ACVIP Plan and the ACVIP Management Plan. The ACVIP handbooks described in this paper provide helpful guidance for these tasks.

## 1 Introduction

A military airworthiness authority qualifies each aircraft according to program-specific requirements derived from military aircraft airworthiness criteria. For the purposes of this paper, our exemplar is the Army Military Airworthiness Certification Criteria (AMACC) [AMACC 2021], but our recommendations may also apply to other organizations, such as the U.S. Navy and the U.S. Air Force, that define their own criteria. The AMACC asserts methods of compliance for contractors to demonstrate that their design satisfies the requirements. Software safety is ensured through compliance with formal development processes and testing of essential elements. The AMACC also calls for verification by analysis to identify and mitigate defects earlier during software design.[1] Despite these safeguards, industry-wide experiences show airworthiness qualification costs continue to rise, because with the increased use of software, there is a corresponding increase in unmitigated software failures due largely to software integration failures [Hansson 2018].

To slow this trend, the U.S. Army has embraced two initiatives. First, the U.S. Department of Defense's (DoD's) Digital Engineering Strategy[2] improves aircraft requirements, design, development, reuse, upgrade, and maintenance through model-based engineering. Using standardized models of software components, the Army can virtually integrate complex aircraft designs before they are built. Second, the Army created the Architecture-Centric Virtual Integration Process (ACVIP) in the Joint Multi-Role (JMR) Science & Technology (S&T) program with model-based analyses to apply as part of model-based virtual integration. ACVIP exposes software integration flaws while the cost to fix them is lower than doing so during the physical realization phase of the program [Boydston 2019]. ACVIP leverages the SAE International Aerospace Standard (AS5506) for Architecture Analysis & Design Language (AADL) [SAE AADL 2017] to target the embedded software and hardware used in the Army's Future Vertical Lift (FVL) aircraft.

To continue to be effective, verification by analysis must adapt to the new model-based reality. Manual methods cannot keep pace with continuously evolving models. Methods that lack a formal foundation will not produce accurate results. Methods that analyze custom or one-off descriptions of the system cannot produce easily validated results. Automating methods wherever possible, applying them to a common, semantically rigorous system model, and combining their results to produce cross-domain validation will generate the objective evidence that the Army airworthiness authority seeks for verification by analysis. Army program offices will need to require ACVIP for verification by analysis for embedded computing system requirements.

The benefits of ACVIP have been proven in recent Army S&T programs,[3] so after a brief introduction to ACVIP in Section 2, we focus on three objectives:

1. Highlight the value that ACVIP analyses will bring to verification by analysis (Section 3).
2. Help the airworthiness authority plan for a transition to ACVIP-generated evidence (Sections 4 and 5).

[1] Section 4.1.2, paragraph (b) of [AMACC 2021] states: [Verification by] Analysis reports shall be submitted prior to the verification test plan package. This approach will facilitate a timely review of the analysis report which, if disapproved, will provide sufficient time to revise the analysis or develop the appropriate test plans to perform the required verification testing.
[2] https://www.acq.osd.mil/se/docs/2018-DES.pdf
[3] We will provide citations in a future revision.

3. Identify transition considerations and next steps (Sections 6 and 7).

## 2 ACVIP in Brief

Model-based development (MBD) bridges the gap between documents and executables (code) with a semantically rich context for expressing system design at a higher level of abstraction than needed to demonstrate machine execution. ACVIP extends MBD to detect flaws too complex for human review but with lower investment than software testing (see Figure 1). Under ACVIP, automated tools examine embedded system architecture models expressed in the AADL [SAE AADL 2017] and identify system specification errors that could lead to software component integration failures. ACVIP is the upward pressure on the model (reflected by the red dotted line in Figure 1) that improves overall design confidence during MBD. Key strategies for ACVIP are to (1) evaluate the evolving design often as new details emerge, (2) discover defects in the current level of design definition, and (3) correct these defects to prepare for the next level of design decomposition, refinement, and sustainment.

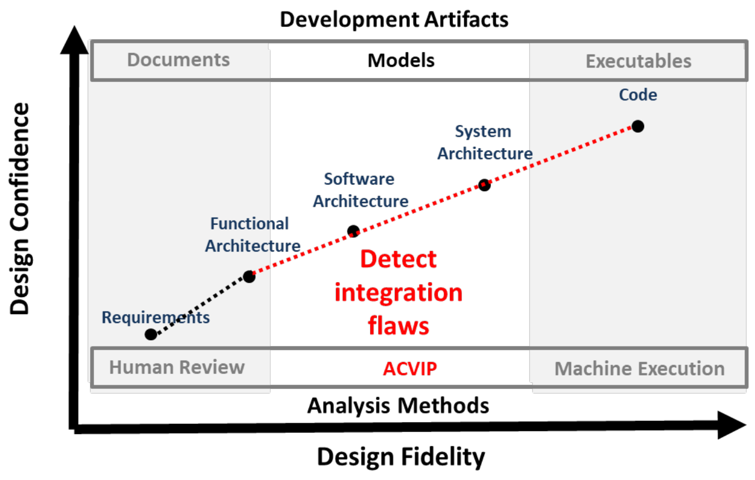


*Figure 1. MBD bridges the gap between requirements and code. ACVIP improves MBD by detecting architectural flaws that could lead to software integration failures and pose a risk to airworthiness qualification.*

ACVIP is supported by the following resources:

1. The *ACVIP Overview Handbook* provides overall motivation, modeling strategies, and workflow guidance [ACVIP Overview 2019].
2. The *ACVIP Acquisition Management Handbook* discusses existing acquisition strategies (including the Modular Open Systems Approach (MOSA), Future Airborne Capability Environment (FACE™)[4], and Comprehensive Architecture Strategy (CAS)), stakeholders, and development milestones and provides a sample workflow [ACVIP Acquisition 2020].
3. The *ACVIP Modeling and Analysis (M&A) Handbook* discusses modeling goals and strategies to support ACVIP and recommends analyses for common development milestones [ACVIP M&A 2021].

[4] FACE™ and logo design are trademarks of The Open Group in the United States and other countries.

4. Model development assets include a FACE to AADL translator, SysML-AADL profile and translator, model templates, and AADL model development environments (e.g., Open Source AADL Tool Environment (OSATE),[5] Ellidiss AADL Inspector[6]).
5. Model analysis assets include OSATE plug-ins and standalone tools.
6. Training offers introductions to both AADL[7] and ACVIP.[8]

According to the *Overview Handbook*, ACVIP enables

1. predictive architecture analysis to facilitate early discovery along multiple dimensions of embedded system properties,
2. incremental development and verification as architecture and detailed designs evolve,
3. standardized model representations to facilitate unambiguous model interchange and an authoritative source of truth (aka single source of truth), and
4. system implementation synthesis based on verified models to minimize residual defects.

## 3 ACVIP Value for the AMACC

To appreciate how ACVIP will enhance airworthiness qualification, we work through an example in Section 3.1 summarizing ACVIP applications for a particular AMACC requirement. A more detailed treatment appears in Table 3 in the appendix. We then discuss in Section 3.2 how ACVIP safety analyses support required safety processes in the AMACC.

### 3.1 Verification by Analysis for an Example AMACC Requirement

AMACC 9.4.2.1 states, “The total system latency for the presentation of primary flight information used for real-time control of an aircraft should not exceed 100 ms.” The methods of compliance for this requirement call for

- verification by demonstration: “The display shall not exhibit flicker that is discernible to the eye.”
- verification by analysis: “Document timing allocations and expected system response times” and “as the system design evolves or is modified, again analyze … based on the updated and refined timing allocations and expected system response times.”
- verification by test: “Test that the latency budgets … are valid for all critical and safety critical tasks and functions.”

ACVIP improves the verification by analysis activity with multiple, complementary analyses on the same model to better define the test cases to be performed later. Contractors perform these analyses iteratively as the model evolves.

Early in the development lifecycle, at the System Requirements Review (SRR) or equivalent milestone, the *ACVIP M&A Handbook* characterizes model content as follows:

[5] https://osate.org/
[6] https://www.ellidiss.com/products/aadl-inspector/
[7] https://www.sei.cmu.edu/education-outreach/courses/course.cfm?courseCode=V40
[8] https://www.adventiumlabs.com/acvip-training

> The SRR model should declare requirements that are allocated to the architecture and its components and are to be verified by analysis of the architecture model… [ACVIP M&A 2021]

With regard to latency analysis at the SRR milestone, the *ACVIP M&A Handbook* states,

> Latency analysis provides information about the timing of data that is flowing through multiple components in a system. Data flows through a system that have latency requirements should be declared in the model using AADL flow declarations. Latency analysis can be applied to functional, software, and hardware viewpoints where information flows are modeled.
> … A simple form of latency analysis is to check consistency between end-to-end flow requirements and subflow requirements derived from them. Analysis that verifies consistency between system latency requirements and derived/allocated subsystem latency requirements may be desired at SRR. [ACVIP M&A 2021]

At the Preliminary Design Review (PDR) or equivalent milestone, the *ACVIP M&A Handbook* characterizes model content as follows:

> The PDR model will be an elaboration of the SRR model that fully identifies software and hardware configuration items and their interfaces. … A PDR model may contain process, subprogram group, and data declarations (software objects); and virtual processor, processor, virtual bus, bus, device, and memory declarations (hardware objects). System objects that have no subcomponents may be used to model either software (if bound to something else) or hardware (if something is bound to them). Abstract objects should be reserved for objects in the environment of use, not in the system being acquired. Subprograms, threads, and thread groups should be modeled where this is needed for a planned structural analysis or when they are separate deliverables, but otherwise this may be unnecessary detail at the structural level of abstraction. [ACVIP M&A 2020]

With regard to latency analysis at the PDR and subsequent milestones, the contractor should repeat the latency analysis performed at SRR but on the more detailed model.

The ACVIP analysis approach satisfies the intent of the AMACC's verification by analysis for this requirement; that is, "as the system design evolves or is modified, again analyze … based on the updated and refined timing allocations and expected system response times" (AMACC 9.4.1.2). But ACVIP goes further than a simple measurement for latency, because ACVIP also exposes potential problems with the latency result itself. A successful latency result carries many assumptions. Observing that software executes within its allotted time, the contractor might assume that external entities do not delay the software, that resource contention or scheduling constraints do not delay execution, and that safety faults or component interaction do not lead to deadlock. The Army airworthiness authority cannot validate these assumptions using the latency result itself, and the latency analysis is incomplete without these validations.

Continuing with the AMACC 9.4.1.2 example, at the SRR milestone the *ACVIP M&A Handbook* also directs the contractor to consider the following analyses whose results could have an adverse effect on the latency result:

1. Interface Behavior Consistency Analysis will detect components whose behaviors could lead to deadlock.

2. Resource Loading Analysis for key performance parameters, such as power and weight, will detect components that fail to operate within their performance envelope.
3. Reliability, Availability, and Failure Analysis will detect components that could fail in particular states.
4. Functional Hazard Analysis, Fault Tree Analysis, Failure Modes and Effects Analysis, and System Theoretic Process Analysis will detect hazards that could block or slow flows.
5. Cross-Domain Analysis will detect the need for cross-domain solutions (e.g., guards), which could increase latency for flows that must traverse those guards.
6. Risk Management Analysis will detect flows of different information criticalities that could interfere with each other.

At the PDR milestone, the *ACVIP M&A Handbook* directs the contractor to consider the following additional analyses whose results could have an adverse effect on the latency result:

1. Failure Modes and Effects Analysis will detect components with insufficient fault handling.
2. Fault Tree Analysis will detect components whose failures are not independent.
3. Reliability Block Diagram Analysis will detect components with unanticipated interdependencies.
4. Markov Analysis will detect components that lack sufficient ability to recover from failures.

In summary, ACVIP analysis generates objective, rigorous, model-based compliance evidence for the specific requirement, but it also validates those results through complementary analyses. Results validation requires a multi-domain approach that includes performance, cybersecurity, and safety considerations. In the next section, we discuss the safety considerations in greater detail.

## 3.2 ACVIP Considerations for Safety

The AMACC discusses the implementation of a comprehensive and robust system safety program that spans the system lifecycle. The aim of the system safety program is to identify any associated system risks/hazards and to eliminate them where possible, or mitigate the risks such that the residual risks are at acceptable levels. The criteria in the AMACC are that a system safety program shall be implemented that mitigates risks/hazards attributed to hardware, software, and human system integration and that the safety program documents and tracks the risks/hazards to the design/modification. The AMACC references several standards—such as SAE Aerospace Recommended Practice (ARP) 4754 systems engineering process, SAE ARP4761 guidelines for safety assessment, and Military Standard (MIL-STD) 882E guidelines for conducting safety assessments—and DO-178C and DO-254 as the means to implement the developmental assurance methodology in the software and complex electronic hardware development processes.

ACVIP, in conjunction with a robust model-centric basis that supports formal analysis of system design and implementation artifacts, can significantly reduce system risks in the safety area. ACVIP allows safety analysis methods to be employed early in the design lifecycle and promotes the ability to perform continuous and iterative analysis of the system architecture that includes both hardware and software. ACVIP can be tailored to work with any system and software development lifecycle model. The approach can be initiated at the requirements analysis phase and can continue through to implementation. As models of the subsystems and component designs evolve, continuous iterative analysis will serve to increase assurance throughout design time. Results from testing of hardware and software artifacts can be compared to model analysis results. The models represent the analyzable blueprint of the hardware–

software system architecture, which can provide a roadmap to sources of discrepancies that can be localized and resolved in a timelier manner.

The AMACC identifies several safety analyses and assessments that are integral in a System Safety Program Plan (SSPP). Table 1 maps safety analysis and assessment artifacts to those identified in the ACVIP handbooks. The ACVIP Support column indicates those assessments, analysis methods, or both that can support those listed in the AMACC. The degrees of support and coverage are determined in part by what is specified in the contractor's ACVIP plan (see Section 5) and SSPP. This mapping is done within the context of an architecture model that incorporates execution platforms, software components, and physical entities. The amount of detail of those components will vary based on the stage in the product lifecycle and the set of requirements that the models and associated analysis are attempting to verify or assess.

*Table 1. Mapping AMACC methods of compliance with ACVIP identified methods. Key: x means that the method applies directly to an AMACC requirement, while * means the method supports AMACC objectives.*

| AMACC System Safety Elements | Applicable Standards | Method of Compliance | ACVIP Support | ACVIP Identified Methods | Supported with AADL Models & EMV2 |
|---|---|---|---|---|---|
| 14.2.1 System Safety Program Plan (SSPP) | MIL-STD-882E | MIL-STD-882E<br>DO-178C<br>DO-254 | x | ACVIP document identifies processes and methods that map to SSPP | *ACVIP Modeling & Analysis Handbook* outlines applicable tools |
| 14.2.2 Preliminary Hazard Analysis (PHA) | MIL-STD-882E | ARP 4761 FHA | x | Model component annotations with hazards, FHA report | x |
| 14.2.3 Functional Hazard Assessment (FHA) | ARP 4761 | ARP 4761 FHA<br>Updated PHA | x | Model component annotations with hazards, FHA report | x |
| 14.2.4 Aircraft Functional Hazard Assessment | ARP 4761 | ARP 4761 FHA<br>Updated FHA to include aircraft functions | x | Model component annotations with hazards, FHA report | x |
| 14.2.5 System-Level Functional Hazard Assessment | ARP 4761 | ARP 4761 FHA<br>Updated FHA<br>Allocate aircraft-level functions to systems | x | Model component annotations with hazards, FHA report | x |
| 14.2.6 Preliminary Aircraft / System Safety Assessment (PASA/PSSA) | ARP 4761 | PASA/PSSA IAW SAE ARP 4761 | x | Model component annotations with hazards, FHA report, aircraft FTA | x |
| 14.2.7 Common Cause Analysis (CCA) | ARP 4761 | CCA IAW SAE ARP 4761, FTA | x | FTA, Minimum Cut Sets, CCA reports | x |
| 14.2.8 Fault Tree Analysis (FTA) | SAE ARP 4761 | Qualitative FTAIAW SAE ARP 4761 | x | FTA | x |
| 14.2.9 System Safety Assessment (SSA) | ARP 4761, Paragraph 3.4<br>ARP 4761, Appendix C | SSA IAW SAE ARP 4761 – Specify verification methods implementation meets design | * | Contributory-models reference verification artifacts and methods | x |
| 14.2.10 Failure Mode, Effects, and Criticality Analysis | ARP 5580 | FMEA IAW SAE ARP 5580 | x | FMEA lite, to be augmented with criticality | x |
| 14.2.11 Safety Assessment Report | MIL-STD-882E, Task 301 | MIL-STD-882E, Task 301 | * | Contributory – include model analysis reports | x |
| 14.2.12 System Safety Hazard Analysis | MIL-STD-882E, Task 205 | MIL-STD-882E, Task 205 | * | Contributory – include model analysis reports | x |
| 14.2.13 Health Hazard Assessment (HHA) | MIL-STD-882E, Task 207 | HHA IAW MIL-STD-882E, Task 207 | NA | NA | NA |

Mapping system-level safety requirements to an architectural solution is the first step in increasing assurance. It is here that Preliminary Functional Hazard Analysis (PFHA) can begin. The hazards can be mapped to architectural components, and design mitigation approaches can be installed. System-level errors that contribute to a hazard can be identified and analyzed for containment, propagation, and mitigation by design or operation. The error types are explicitly defined in AADL, attached to architectural components in the model, and then integrated and propagated to understand the effects.

AADL, in conjunction with the Error Model Annex, Version 2 (EMV2), can directly support the safety analysis methods identified in the AMACC. The analysis methods that are supported by EMV2 include Functional Hazard Assessment (FHA), Failure Mode and Effects Analysis (FMEA), Fault Tree Analysis (FTA), and Common Cause Analysis (CCA). AADL supports an FHA by annotating a model of the system with Hazards property values. This can be done on a functional architecture, a physical architecture, or the deployment of a functional architecture to a physical architecture. It can also be done on a task and communication architecture deployed on a hardware platform. FHA can be performed in a generic EMV2 format, in SAE ARP4761 format, or in MIL-STD-882 format. Key words and terminology particular to each of the safety formats are used in the modeling and analysis. In OSATE, fault impact analysis is equivalent to FMEA. It looks at the impact of fault occurrences on a system, which is determined through forward reasoning from an error source.

FTA is a form of deductive fault impact analysis. Fault occurrences are identified as contributors to a critical failure effect, such as an accident through backward reasoning. FTA capability in OSATE supports the generation of fault contributor traces with identification of dependent events, such as component failures that affect multiple other components, fault trees that are flattened and transformed fault traces with occurrence probability calculation, and minimal cut sets with occurrence probability calculation. It supports event and gate types defined in the Nuclear Regulatory Commission Fault Tree Handbook (NUREG-0492). A more detailed description of the methods and their applicability to the safety process in general is contained in the ACVIP M&A Handbook. Furthermore, tools within certain development environments, such as OSATE and Collins Aerospace's publicly available Assumed Guarantees Reasoning and Evaluation Environment (AGREE), and languages like Resolute allow users to develop their own analysis capability. For example, a system may have multiple confidentiality categories of data, which can only be accessed by components with the same or higher confidentiality level. The user can write additional model analysis in AADL's Resolute or AGREE languages to ensure that the system enforces the confidentiality rules.

Table 2 shows the mapping of safety process elements related to SAE ARP4761 and MIL-STD-882E and the constructs within EMV2 that are used to support those process elements. For more detail about the error modeling and analysis capability, see [Feiler 2016a], [Feiler 2016b], and [Delange 2014].

Table 2. ARP4761 Process Elements and Supporting AADL Error Model Constructs

| AADL Error Model (EMV2) Constructs | | ARP4761 & MIL-STD 882E Process Elements: Functional Hazard Assessment (FHA) | Fault Tree Analysis (FTA) | Failure Mode and Effect Analysis (FMEA) Fault Impact Analysis | Markov Analysis | Dependence Diagram (Reliability Block Diagram) |
|---|---|---|---|---|---|---|
| Error flows | Error propagation | x | x | x | x | x |
| | Error source | x | x | x | | |
| | Error path | | | x | | |
| | Error sink | | x | x | | |
| Error behavior | Error states | | x | | x | x |
| | Error transitions | | x | | x | x |
| | Error events | x | x | x | | x |
| | Composite error model | | x | | | x |
| Properties | *Hazards* property | x | | | | |
| | *OccurrenceDistribution* property | | | | x | x |

## 4 How ACVIP Supports Army Objectives

Army Regulation (AR) 70-62 provides the charter for the role of assessing the airworthiness of Army aircraft systems. It is established by the order of the Secretary of the Army and lists a number of responsibilities to support organic airworthiness functions. These include establishing airworthiness requirements, assessment of design compliance from verification, and airworthiness approvals. The Commanding General, U.S. Army Aviation and Missile Command (AMCOM), will “develop and implement a fully coordinated program for oversight of airworthiness qualification for aircraft, subsystems, components and allied equipment” [Army 2016]. Furthermore, per AR70-62 the AMCOM Commanding General will “ensure that systems safety engineering and management principles, criteria, and techniques are applied to achieve optimally low safety risk.” ACVIP provides strong analytical support for verification of embedded system non-functional requirements as well as the correctness of integrated interaction behavior across components.

The Defense Advanced Research Projects Agency (DARPA), the Aerospace Vehicle Systems Institute’s (AVSI’s) System Architecture Virtual Integration (SAVI) project, and the U.S. Army Combat Capabilities Development Command Aviation and Missile Center (DEVCOM AvMC) Joint Multi-Role (JMR) Mission

System Architecture Demonstration (MSAD) Science and Technology (S&T) project have already developed the language, method, processes, and tools and carried them through S&T experiments with Army contractors to mature the tools and start the transition. Contractors who use a standard embedded systems architecture language for components and architecture make the Army airworthiness authority's job of incremental design verification through analysis easier. ACVIP requirements are part of major Amy programs, so it is expected that the airworthiness authority will get the opportunity to evaluate ACVIP analysis of contractor models. With ACVIP analyses, the airworthiness authority will gain a significant capability to understand and verify their systems more rapidly.

ACVIP using AADL provides a standard, analyzable specification with clear semantics defined for each foundational component and interaction used in real-time embedded systems. It also provides the meaning of standard properties and the dynamics of system execution. It provides a compositional approach to system analysis, based on component boundaries, defined hierarchy rules for system composition, and execution behaviors to enable scalable formal analysis. Because of this standardized meaning, rather than tool-specific or vendor-customized semantics, the work load to understand the architecture is significantly reduced by this common language and it can be analyzed throughout the system lifecycle. This translates into reduced complexity for the airworthiness authority engineer verifying a model. It also translates into increased coverage of requirements through formal analysis to complement testing.

As the aviation industry moves toward new model-based engineering (MBE) and model-based systems engineering (MBSE) technology, personnel will need initial mentoring and continual training to use the evolving technology effectively in the qualification role. Assessing design compliance from verification will involve analysis of engineering models to a greater degree, including models that predict embedded system performance to requirements for airworthiness. AR 70-62 provides that authority. Therefore, the airworthiness authority can request that this training be made available through leadership.

Aviation program executive offices (PEOs) and their respective program management offices (PMOs) will "program for development of the system design, design documentation, verification, and life cycle airworthiness activities to generate data in accordance with approved airworthiness requirements"(AR 70-62 [Army 2016]). PMs must ensure that the information needed for assessment is provided. The PEO and its PMs are responsible for determining that a program should leverage what is now available through MBSE and MBE technology enhancements to provide the design documentation, analysis requirements, architecture specifications, and development processes needed to get to an optimally low safety risk.

When the qualified aircraft is modified under AR 70-62 requirements for the airworthiness authority, according to Section 2-1 c, "All modifications impacting airworthiness will subject the aircraft system, subsystem, component or allied equipment to requalification" [Army 2016]. So the airworthiness authority is required to assess the impact of change on airworthiness and require requalification if necessary. This task will likely require some evidence of the probable impact, given the cost of requalification. However, even small changes can have significant impact on the safety of an embedded software-driven system.

ACVIP provides an automated means to quickly assess the impact of change given AADL specifications and contractor-delivered analyses of the system. These can be used to run trade space alternatives to minimize airworthiness impact. Given an anticipated change, virtual integration of the system can provide a new capability for this assessment prior to the change across a set of critical analyses that all results must stay within required limits to meet airworthiness criteria. In fact, different changes that have the desired impact on the system can be evaluated to see which has the least impact on requalification. Detecting side effects is typically a daunting task, often left to testing to discover. But the test–fix cycle can lead to circular failures. Test-driven changes have no ability to predict side effects, often resulting in changes that trigger new side effects, invalidating previously tested and qualified aspects of the system. Predicting side effects reduces the cost of requalification by minimizing the impact of the selected changes. Through multi-dimensional integrated analyses, ACVIP provides a way to predict the effects of change across the architecture, supports decisions about whether the change can be made without requalification, or can be used to minimize the impact of the change on requalification.

# 5 How ACVIP Can Fit into the Airworthiness Qualification Workflow

In this section, we present one possible scenario for augmenting the airworthiness assessment process with ACVIP activities and analyses. Augmenting a system development process with ACVIP activities changes the effort profile of the process. Defects are detected early in the process, reducing the effort required later in the process.

Per recommendation from the *ACVIP Acquisition Handbook*, the government program office initiates the workflow by producing an ACVIP Plan that outlines the ACVIP information that is included at the major milestones and reviews for the program. (See interaction 4 in Figure 2.) The development of this plan provides the opportunity for tailoring the overall program plan, which becomes Government Furnished Information (GFI) for the program vendors. The contractors, in turn, propose ACVIP Management Plans that respond to the ACVIP Plan to document the details of how the contractor’s development process will address the issues raised in the ACVIP Plan. (See interaction 7 in Figure 2.) The ACVIP Plan and ACVIP Management Plan should support the Airworthiness Qualification Plan (AQP) in providing insight into system development. The PM will need to coordinate with the airworthiness authority to ensure coverage of AQP requirements.

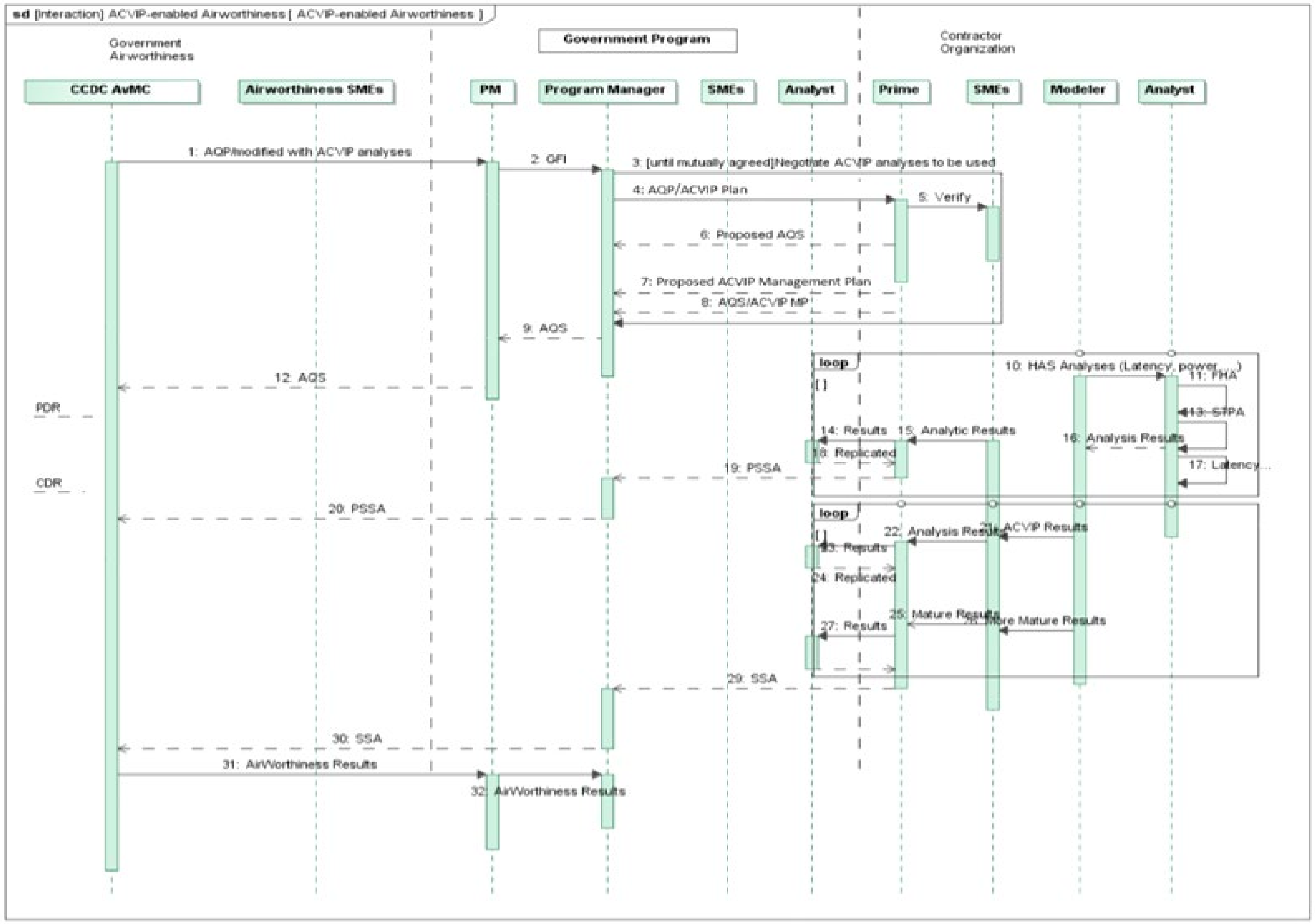


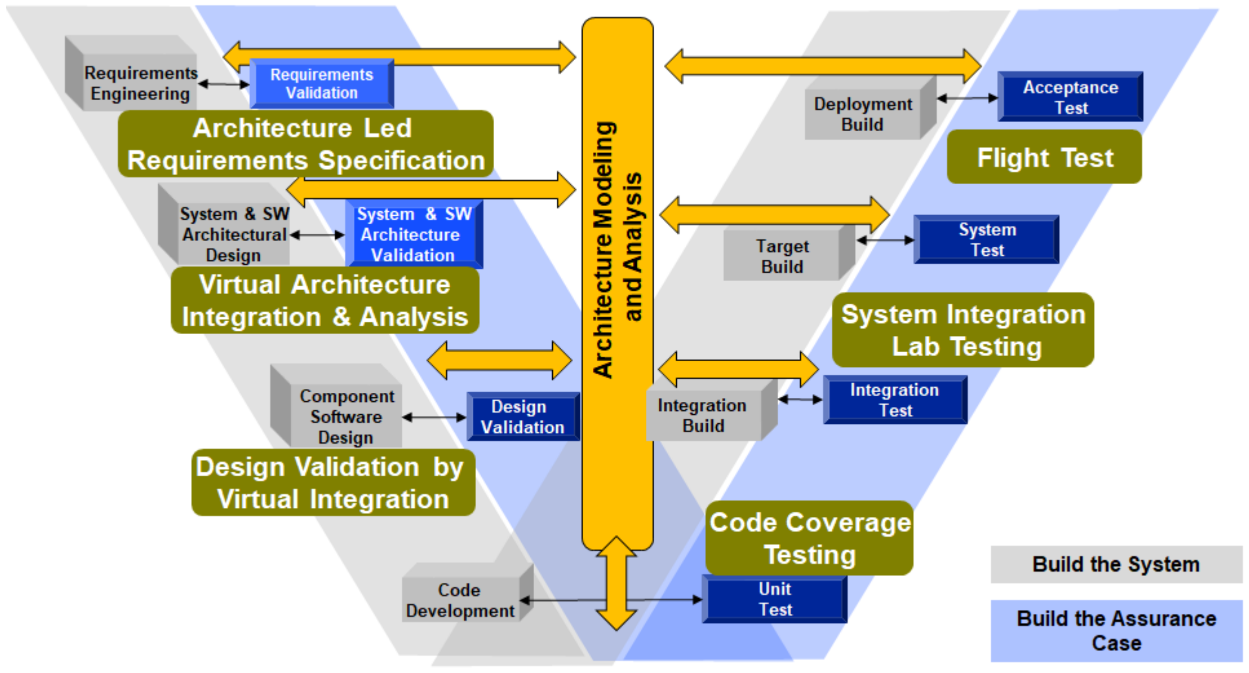
*Figure 2. A role for ACVIP in the airworthiness qualification workflow. The loops on the right side repeat for each development milestone (not shown). (Source: SEI)*

The airworthiness authority with jurisdiction over this program imposes specific safety requirements—including functional, non-functional, and process—on the new development effort. System engineers developing the Concept of Operations (CONOPS), and subject-matter experts beginning safety and cybersecurity assessments, bring constraints as well as experience-based estimates of system parameters that serve as the basis for very early analytic AADL models. These requirements are in whole, or in part, represented in model form (such as a SysML requirements diagram or specification tables).

Activities such as requirement flow-down and multi-fidelity modeling from the ACVIP workflow are integrated with a tailored version of the DoD Instruction (DoDI) 5000.02 Major Capability Acquisition pathway. For example, the workflow might be tailored to include a process fragment for developing the PSSA. In this process, fragment safety requirements are defined at both the system and component levels and FHA is initiated. The hazards found by the FHA can be captured in the error annex portions of the AADL model. The safety requirements can be captured in the model or in an external tool and imported into the model along with other categories of requirements.

The results of these system-level analyses support software engineers during the software development iterations of the Software Acquisition pathway. ACVIP is most effective when the software engineers can model scenarios and investigate alternative designs using AADL-based analyses such as resource utilization (e.g., bus loading, processor loading, memory loading) and resource budgeting (e.g., power, weight) to provide quantitative evidence that is used to guide design changes. As development proceeds, the developers modify the design model, including updating the values of quality attributes. The analyses can be repeatedly invoked to monitor the impact of design changes on the architecture, such as those defined in the AMACC in the Human Engineering section. Values of many of these attributes are also required for the system metric (SM) report, SM Report IAW DI-MISC-80508. The verification plan links the model-based analyses to code-based tests, which ensure that constraints verified in the models have been completely, correctly, and consistently implemented in the code. For example, a flow in the model, which has been subjected to latency analyses for error detection and correction, should be the source of test cases that time the actual execution.

ACVIP also supports the integration team in creating a virtually integrated, model-based system specification and design by combining the component designs of the individual software engineering teams. The AMACC's PSSA process identifies points in the process at which individual components are analyzed and points at which the integrated system is evaluated. In Figure 3, the Verification and Validation (V&V) model shows this process phase corresponding to the Component Design phase.


Requirements Engineering
Requirements Validation
Architecture Led Requirements Specification
System & SW Architectural Design
System & SW Architecture Validation
Virtual Architecture Integration & Analysis
Component Software Design
Design Validation
Design Validation by Virtual Integration
Code Development
Architecture Modeling and Analysis
Deployment Build
Acceptance Test
Flight Test
Target Build
System Test
System Integration Lab Testing
Integration Build
Integration Test
Code Coverage Testing
Unit Test
Build the System
Build the Assurance Case

ACVIP works best if there are clear workflows from analytic models to executable code. Earlier we discussed the refinement of a modeled specification from abstract components to detailed design. This refinement continues from detailed design into implemented code. Whether generating code manually or through automated means, ACVIP provides the most value when used with a rigorous workflow that synchronizes models and code. Of course, any generation of safety-critical code either requires the qualification of the tool or manual verification of the generated code to low-level requirements [Adventium 2021b].

## 6 Transition Considerations for Adopting ACVIP

This section highlights some development, analysis, and process changes for the airworthiness authority and PM to consider in the transition to ACVIP-generated evidence for verification by analysis.

**Set Expectations for the Contractor.** ACVIP-enabled MBD changes the qualification process and its artifacts of importance. Models largely replace textual documents as the central design artifacts (although textual documents can be incorporated into the models as necessary) and facilitate evaluation of the design by multiple stakeholders that have different perspectives and concerns. Modeling activities should begin early in the contractor's development lifecycle to help with requirements verification and refinement and to establish the system architecture that serves as the blueprint for downstream software and hardware development and implementation.

**Require Models of the Software.** Software provides a significant amount of the system functionality, and the software architecture model specifies how the software functions within the context of the entire system. In essence the software is modeled by expressing its functional interface, component classification, and quality attributes as component properties within a system structure. Expressing system software design and allocation to processing hardware is a key activity within ACVIP. The ability to make design choices early in the development lifecycle and quantitatively analyze the model establishes a design baseline in which system properties in the areas of performance, safety, and cybersecurity can be substantiated. The cadence of model updates and analysis should be established as part of the ACVIP Plan. In the early stages of development, the model can be fairly abstract but would capture the structure of the system, software patterns of communication, data specification, tasking and communication, and allocation to execution hardware.

**Define a Model-Centric Workflow.** From an organizational perspective, the system model encompasses a lot of information that dictates the design of the system. The model should be a collection of requirements, standards, component descriptions, interfaces, connections, software, computing platforms, communication buses, sensors, controlled elements, and so forth. Together the pieces of this collection represent the embedded system specification that governs system development and is eventually transformed into the implemented system. The model (the software, hardware, and relationships among them) contains the design that is intended to meet the system requirements. The model should contain information and properties from many different domains such as safety and cybersecurity. The ability to view the modeled system and conduct analysis from multiple domain viewpoints should increase assurance that the resultant system will meet those domain-specific requirements. All models and analysis results should be under version control and configuration management to support the authoritative source of truth.

**Encourage Predictive Analysis.** Model analysis results are an indication, early in the development lifecycle, of the feasibility of meeting requirements. In this work, we have identified areas in the AMACC where this approach can add value and increase confidence that the design will satisfy the criteria. In the transformation from design to implementation, issues can arise that may invalidate some of the property values contained in the design or the structure of the design itself. MBD provides a direct way to incorporate into the model known truths from the physical realization and gauge their effects across the entire system. This approach provides insight as to what design or implementation changes need to occur to help ensure that the as-built system will pass the airworthiness qualification criteria.

**Require Consistency across Modeling Languages.** No one modeling language will address every concern of system representation and verification at every level in the design and implementation hierarchy. Modeling language semantics must be precise enough to allow meaningful analysis, and component elements and their properties represented in multiple modeling languages must have identical semantics. A program must plan for transitioning components and property values from one modeling language to another. For example, the latency definition of a device should be the same in both SysML and AADL to have meaningful end-to-end signal latencies at the embedded system architecture level. This goes beyond things such as consistency of units. For example, in the case of a smart sensor, latency could have multiple semantic interpretations. Sensor latency could be defined as reading the sensor, processing the value, and making the value available at its interface, or only the time that it is present at the interface, ignoring the physical aspect such as inertia. Preserving semantic consistency will ensure that analysis results are correctly interpreted. This aspect also becomes important in verification of the implemented system as signal and timing measurements must be consistent with the semantics of the AADL model. The job of ensuring semantic consistency among tools would most likely be handled by a system engineering group or ACVIP design team. To ensure comprehensiveness of the integrated model across multiple domains, the team should include system engineering, software engineering, safety engineering, cybersecurity, and qualification experts. Perspectives from people in each of these domains can contribute to and assist with system-wide trade-offs of the architecture to accomplish continued integration feasibility.

**Adapt Verification by Test.** As we noted in Section 3.1, verification by analysis may permit better refinement in the use of verification by test, and models can serve as the basis for test case generation that will augment current testing approaches. For example, interface specification of model components could be used to generate test vectors for proper handling of variable range, corner cases, and other parameters. Models could also specify the architectural properties to verify at runtime. Tools could be developed that would use model properties to generate software probes into the target system to acquire the verification data.

**Train Stakeholders.** Training and leveraging the experience of researchers and practitioners is an important aspect of successfully transitioning to and employing of ACVIP. System engineers need to develop an understanding of both the interfaces and overlap between system and embedded software modeling. Software-intensive systems provide a significant amount of functionality and also contribute to overall quality attributes of systems such as performance, fault tolerance, and safety. System engineers need to consider architectural characteristics in system modeling activities such that those architecturally significant properties and characteristics can be incorporated into architectural models for analysis. Conversely, embedded systems architects need to know the relevant system engineering artifacts that will flow to the embedded architecture being designed. In addition, important system-wide requirements that have been decomposed must also be incorporated into the architecture design of the embedded system as the architecture model is the answer to meeting the requirements.

One aspect of training for system architects and engineers, and in particular embedded system architects and engineers, involves the art of modeling. Model organization, access, and component usage must be well planned. In addition, the continuous integration aspect of periodically rerunning model verification analysis suites needs to be documented. Another aspect of the art is knowing how much detail to put into which components. In some respects, modeling components abstractly with certain properties will be sufficient for analysis purposes. For example, identifying and modeling critical data and control flows instead of potentially all the flows in a component would be sufficient for some verification criteria. There are situations where more detail is required, for example, generating glue code for a set of threads that call a number of software modules and system scheduling by the processor. The incorporation of more software modules would tend to minimize faults injected by manual integration.

Software engineers from both the government side of the project and the development side (contractors) will need to develop an understanding of model-based embedded system architecting. The contractors supply models to be reviewed by the government. Their understanding will need to be deep in order to design the models. They will need to develop an understanding of embedded system architecting and the value brought by modeling and analysis of software and hardware at a level of abstraction that is higher than code, and what value is brought by modeling at an abstract level. Software engineers can use formal representation of a communication pattern, data locality (local vs. global) and access methods, thread representations and processor load capability, and other model elements to analyze for a number of system characteristics, such as performance, resource utilization, and error propagation. Analysis at this level of detail ensures that system properties can be met. It serves to reduce errors that manifest themselves when code products are integrated and then reveal that certain architectural assumptions have been violated. Analysis of the architecture provides assurance through the design lifecycle that the system will meet criteria specified in the AMACC.

The airworthiness authority will need to understand MBD and ACVIP in order to properly evaluate and leverage ACVIP-produced results during verification by analysis. SEI and Adventium offer training on both of these concepts (see Section 2). With this background, the airworthiness authority can relate the analysis results directly to requirements (improving on traditional requirements tracing), can apply the results to highlight requirement and design deficiencies, and can leverage the results to better define the downstream verification of actual implementations and simulations. Verification can be used against requirements as well as architecture analysis. Test results on the implementation that do not meet qualification acceptance criteria can be used to localize where defects may have been injected during the translation of models to code. The models can be used to gain insight as to where the problems exist. If architecture changes are necessary to resolve an issue, the airworthiness authority can evaluate "what if" analysis results the contractor generates using the model, gauge the appropriateness of proposed solutions, and explore alternatives.

# 7 Next Steps

In this paper, we outlined how ACVIP can enhance the AMACC's verification by analysis. In this section, we identify the steps to achieve that objective.

To begin, the PM and contractors should use the AQS and ACVIP Management Plan to propose the method of Verification for the airworthiness requirements that are identified in the AQP. Those that fit under Verification by Analysis are the likely candidates that will benefit from an ACVIP approach. (See the appendix for a few more examples.) With their experience in the airworthiness qualification process, the airworthiness authority should suggest risks that ACVIP could help mitigate. Pragmatically, it makes sense to pare down the requirements to a manageable set to pursue.

Next, the PM and contractors should use the AQS and ACVIP Management Plan to propose the appropriate ACVIP analyses for these requirements. The airworthiness authority needs a verification plan that includes model validation, model analysis techniques, and the relationship to implemented hardware and software, and that ties in testing approaches of the implemented system. This activity would essentially establish the game plan for verification of the models and the implemented system. The model verification analysis should largely map to testing of the implemented system.

Finally, the PM and contractors should define levels of model maturity that align with program reviews, established model maturity milestones, or both. These levels of maturity will determine the analyses to perform at each program milestone, which are key inputs for the government supplied ACVIP Plan. The maturity and adequacy of the models received from the contractors could be graded at various phases of the program and fed back to the performer for improvement.

In closing, a significant effort to develop, exercise and mature ACVIP was executed on the U.S. Army's Joint Multi-Role Mission Systems Architecture Demonstrations from 2013-2020 including the:

- Joint Common Architecture Demonstration [US Army JCA Demo 2016][Adventium JCA 2015] [SEI JCA 2015a] [SEI JCA 2015b],
- Architecture Implementation Process Demonstrations [US Army AIPD 2018] [Adventium AIPD 2017] [SEI AIPD 2018]

- Capstone Demonstration [US Army Capstone 2021] [Adventium Capstone 2021] [SEI Capstone 2021]

ACVIP is now transitioning to FVL programs as a requirement to mitigate late phase integration issues in embedded systems.

On the longer time horizon, analysis and system building capabilities continue to be developed and should be matured for inclusion to leverage advanced ACVIP research and development. Formal method techniques utilizing ACVIP have been developed, exercised and matured by the Defense Advanced Research Projects Agency (DARPA) for the High Assurance Cyber Military Systems [DARPA HACMS 2017] [Collins SMACCM 2017] and Cyber Assured Systems Engineering [DARPA CASE 2022] programs. These DARPA S&T programs have provided assured trusted build techniques and tools that could be applied not only to cybersecurity certification but also for airworthiness qualification. [Loonwerks HACMS CASE 2021].

## 8 Bibliography

[ACVIP Acquisition 2020] Software Engineering Institute. Architecture-Centric Virtual Integration Process (ACVIP) Acquisition Management Handbook. CMU/SEI-2020-SR-021-Restricted. August 2020.

[ACVIP M&A 2021] Adventium Labs. Architecture-Centric Virtual Integration Process (ACVIP) Handbooks – Modeling & Analysis with the Architecture Analysis & Design Language (AADL). March 2021. https://www.adventiumlabs.com/sites/default/files/documents/ACVIP-Modeling-%26-Analysis-Handbook_Mar2021_DistA.pdf

[ACVIP Overview 2019] Software Engineering Institute. Architecture-Centric Virtual Integration Process (ACVIP) Handbooks – Overview with the Architecture Analysis & Design Language (AADL). April 2019.

[Adventium 2021a] Adventium Labs. ACVIP Training. 2021. https://www.adventiumlabs.com/acvip-training

[Adventium 2021b] Adventium Labs. Airworthiness Qualification of ACVIP Tools." August 2021. IN DRAFT.

[AMACC 2021] U.S. Army Combat Capabilities Development Command Aviation & Missile Center (DEVCOM AvMC). Army Military Airworthiness Certification Criteria (AMACC) Revision A Change 2 (C2). April 9, 2021. https://www.avmc.army.mil/Directorates/SRD/TechDataMgmt/

[Army 2016] Department of the Army. Airworthiness of Aircraft Systems. Army Regulation 70–62. May 2016. https://armypubs.army.mil/ProductMaps/PubForm/Details.aspx?PUB_ID=1000692

[Boydston 2019] Boydston, Alex K.; Feiler, Peter H.; Vestal, Steve; & Lewis, Bruce. Architecture Centric Virtual Integration Process (ACVIP): A Key Component for the DoD Digital Engineering Strategy. 22nd Annual Systems and Mission Engineering Conference. September 2019. https://www.adventiumlabs.com/publication/architecture-centric-virtual-integration-process-acvip-key-component-dod-digital

[Delange 2014] Delange, Julien; Feiler, Peter; Gluch, David; & Hudak, John. AADL Fault Modeling and Analysis Within an ARP4761 Safety Assessment. CMU/SEI-2014-TR-020 . Software Engineering Institute,

Carnegie Mellon University. 2014.
https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=311884

[Feiler 2015] Feiler, Peter H. & Hudak, John J. Potential System Integration Issues in the Joint Multi-Role (JMR) Joint Common Architecture (JCA) Demonstration System. CMU/SEI-2015-SR-030. Software Engineering Institute, Carnegie Mellon University. December, 2015.
https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=447176

[Feiler 2015a] Feiler, Peter; Weinstock, Charles; Goodenough, John; Delange, Julien; Klein, Ari; & Ernst, Neil. Improving Quality Using Architecture Fault Analysis with Confidence Arguments. CMU/SEI-2015-TR-006. Software Engineering Institute, Carnegie Mellon University. 2015.
http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=435051

[Feiler 2016a] Feiler, Peter; Hudak, John; Delange, Julien; & Gluch, David. Architecture Fault Modeling and Analysis with the Error Model Annex, Version 2. CMU/SEI-2016-TR-009. Software Engineering Institute, Carnegie Mellon University. 2016. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=464380

[Feiler 2016b] Feiler, Peter H. & Delange, Julien. Automated Fault Tree Analysis from AADL Models. ACM High Integrity Language Technology International Workshop on Model-Based Development and Contract-Based Programming (HILT). Pittsburgh, PA. 6-7 October 2016.
https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=499346

[Hansson 2018] Hannsson, Feiler and Helton. ROI Analysis of the System Architecture Virtual Integration Initiative. SEI Technical Report CMU/SEI-2018-TR-002, 2018.
https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=517157

[SAE AADL 2017] Society of Automotive Engineers. Architectural Analysis and Design Language. SAE AS 5506C-2017. 2017.

[SEI 2021] Software Engineering Institute. Modeling System Architectures Using the Architecture Analysis and Design Language (AADL) – eLearning. 2021. https://www.sei.cmu.edu/education-outreach/courses/course.cfm?courseCode=V40

[US Army JCA Demo 2016] Wigginton, Joint Common Architecture Demonstration (JCA Demo) Government Final Report, Technical Report RDMR-AD-16-01, DTIC Library, July 2016.

[Adventium JCA 2015] Vestal. Joint Common Architecture Demonstration Shadow Architecture Centric Virtual Integration Process Final Report, 9 October 2015. DTIC Library.

[SEI JCA 2015a] Feiler. Architecture-Led Safety Analysis of the Joint Multi-Role (JMR) Joint Common Architecture (JCA) Demonstration System, December 2015. DTIC Library.

[SEI JCA 2015b] Feiler. Requirements and Architecture Specification of the JMR JCA Demonstration System, December 2015. DTIC Library.

[US Army AIPD 2018] Jacobs, Boydston, Dennis, Salvetti, Johnson, Yost. Architecture Implementation Process Demonstrations (AIPD) Government Final Report. March 2018. DTIC Library.

[Adventium AIPD 2017] Vestal, Joint Multi-Role Helicopter Missions Systems Architecture Demonstration Architecture Implementation Process Demonstrations Support Final Report. 1 Dec 2017. DTIC Library.

[SEI AIPD 2018] Cohen, Hudak. Architecture Implementation Process Demonstrations (AIPD) Final Report. April 2018. DTIC Library.

[US Army Capstone 2021] Jacobs, Boydston, Ba, Rhamy, Davis, Padilla, Roberson, Dennis, Kinch, Yost. Joint Multi-Role Mission System Architecture Demonstration – Capstone Demonstration Government Final Report. May 2021.

[Adventium Capstone 2021] Vestal. JMR MSAD Capstone Model-Based Engineering Support. 30 March 2021.

[SEI Capstone 2021] Cohen, Anderson, Nichols, Schenker, McGregor, Hudak. ACVIP in the Capstone Science & Technology Effort. June 2021, CMU/SEI-2021-SR-018.

[DARPA HACMS 2017] Richards. High Assurance Cyber Military Systems (HACMS) (Archived), https://www.darpa.mil/program/high-assurance-cyber-military-systems.

[Collins SMACCM 2017] Cofer, Backes, Gacek, DaCosta, Whalen, Juz, Klein, Heiser, Pike, Foltzer, Podhradsky, Stuart, Grahan, Wilson. Secure Mathematically-Assured Composition of Control Models, AD1039782, 2017-09-27.

[DARPA CASE 2022] Martin. Cyber Assured Systems Engineering (CASE), 2022. https://www.darpa.mil/program/cyber-assured-systems-engineering

[Loonwerks HACMS CASE 2021] Loonwerks, CASE: Cyber Assured Systems Engineering, 2021. http://loonwerks.com/projects/case.html

## Appendix

Table 3 presents sample AMACC requirements that the PM and contractors might find suitable for ACVIP-driven verification by analysis. The first column, “Criteria,” contains the text from the AMACC; the second column, “Analysis Requirement,” contains text from the requirement’s Method of Compliance; and the third column, “Model?” indicates whether the PM and contractors expect the model to support this analysis. The next four columns indicate analysis activities to perform at different stages of model maturity; these stages might correspond to the development milestones in the *ACVIP M&A Handbook*. The final column indicates the value obtained by this analysis.

*Table 3. ACVIP multi-domain analysis applied to AMACC examples.*

| **Criteria [AMACC 2021]** | **Analysis Requirement** | **Model?** | **Model as a Black Box** | **Define Functional Architecture and Modes** | **Refine to Processes and Threads** | **Bind to Execution Platforms** | **Value Type** |
|---|---|---|---|---|---|---|---|
| 9.4.2.1. The aircraft system, subsystems and components critical tasks and functions shall not create visual distraction or impact human response dynamics. | The information update rate and screen refresh rate shall provide for symbols to move smoothly and not flicker. | Yes | Capture flows that could cause flicker, specify budgets and analyze impact of budgets, include legacy impacts with more definition | Allocate jitter on latency to subsystem level, analyze impact | Latency with respect to thread worst-case execution time range estimates impacting jitter, capture detailed connection flows, analyze | Schedule min/max wrt 653 Frame miss, latency impact, processor, bus loading impact, analyze | Evidence insight |
| | The total system latency for the presentation of primary flight information used for real-time control of an aircraft should not exceed 100 ms. | Yes | Latency [4.5.8] - Model flow budgets | Latency [4.6.5] - will subsystems satisfy budgets in all modes? | Latency [4.6.5] - will threads satisfy budgets? | Schedulability [4.7.4] - can we schedule threads to satisfy budgets? Frame rates, processor rates, bindings, OS delays | Risk insight, comprehensiveness |
| 9.4.2.2. The aircraft system shall have no timing or latency anomalies that degrade safe operation. | Analyze and test that all normal, backup, and emergency modes of operation are safe for the integrated system. | Yes, but define "safe operation" | Capture required mode states and safety requirements in BB. | Model modes (normal, backup, and emergency); validate against "safe operation." Define mode switching times, connections, and triggers. Allocate and analyze. | Refine all related property values; validate each mode to requirements. | Refine values; schedule, validate, build embedded prototype to improve timing estimates and demonstrate | Comprehensiveness |
| | Timing (e.g., priorities, bounded timelines) and latency anomalies for normal, backup, and emergency modes shall be analyzed, tested | Yes, but define "safe operation" | | Model modes normal, backup, and emergency; validate against budgets. | Refine values; validate | Refine values; schedule, validate, build embedded prototype to test all mode changes, measure timing, adjust model or analysis if required. | Risk insight |
| 9.4.2.3. The aircraft system shall have control functions, payloads, and ground systems latencies and synchronizations that ensure usability and safe operation. | The vehicle control function integrated with any payload or ground functions shall not induce latencies that result in flying and handling qualities worse than Level I (100 ms) | Yes | Latency [4.5.8] - Model flow budgets | Latency [4.6.5] - will subsystems satisfy budgets? | Latency [4.6.5] - will threads satisfy budgets? | Schedulability [4.7.4] - can we schedule threads to satisfy budgets? | Design insight |
| | The Vehicle Control Function (VCF) integration with the payload or the control station is evaluated for safe and adequate timing [and] synchronization rates. | Yes | Latency [4.5.8] - Model flow budgets<br>FHA [4.5.10] Identify system-level hazards | Latency [4.5.8] - will subsystems satisfy budgets?<br>PSSA [4.5.10, 4.5.9] composed of FMEA [4,7,5], FTA [4.7.6], CCA, | Latency [4.5.8] - will threads satisfy budgets?<br>SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated | Latency [4.5.8] - can threads be scheduled to satisfy budgets?<br>SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], | Comprehensiveness |

| Criteria [AMACC 2021] | Analysis Requirement | Model? | Model as a Black Box | Define Functional Architecture and Modes | Refine to Processes and Threads | Bind to Execution Platforms | Value Type |
|---|---|---|---|---|---|---|---|
| | | | Identify synchronization issue error effects and mitigations | RBD [4.7.7], Markov analysis [4.7.8], SW faults identified, mitigated? | | SW+HW faults identified, mitigated | |
| | The Vehicle Control Function (VCF) integration with the payload or the control station is evaluated for synchronization rates. | Yes | 1. Latency [4.5.8] - Model flows, assign latency budgets<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify component states, behavior<br>3. FHA [4.5.10] Identify system-level hazards<br>4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Specify interface type, data, units, behavior | 1. Latency [4.5.8] - System decomposition, hierarchical flows. Component flows latency <= system latency requirements? Evaluate under synchronous/asynchronous communication.<br>2.Behavioral Consistency [4.5.6, 4.6.3, 4.7.3]<br>3. PSSA [4.5.10, 4.5.9] composed of FMEA [4,7,5], FTA [4.7.6], CCA, RBD [4.7.7], Markov analysis [4.7.8], SW faults identified, mitigated?<br>4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent between connected components? | 1. Latency [4.5.8] - System thread and& subprogram models, hierarchical flows. System flows latency less than equal to system latency requirements? Evaluate under synchronous/asynchronous communication.<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify thread states, behavior (thread deadlocks?)<br>3. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated.<br>4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent between connected components? | 1. Latency [4.5.8] - Flows through devices, busses, hierarchical flows. System flows latency less than equal to system latency requirements? Evaluate under synchronous/asynchronous communication.<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify processor modes<br>3. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated | Design insight |
| | The Vehicle Control Function (VCF) integration with the payload or the control station is evaluated for instruction set architecture. | No | | | | | Design insight |
| | The Vehicle Control Function (VCF) integration with the payload or the control station is evaluated for misinterpretation of instructions or data. | No | | | | | Pedagogy |
| | The Vehicle Control Function (VCF) integration with the payload or the control station is evaluated for degraded data link. | Yes | 1. Latency [4.5.8] - Model flows, assign latency budgets<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify component states, behavior<br>3. FHA [4.5.10] Identify system-level hazards<br>4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] | 1. Latency [4.5.8] - System decomposition, hierarchical flows. Component flows latency <= system latency requirements? Evaluate under synchronous/asynchronous communication.<br>2.Behavioral Consistency [4.5.6, 4.6.3, 4.7.3]<br>3. PSSA [4.5.10, 4.5.9] composed of FMEA [4,7,5], FTA [4.7.6], CCA, | 1. Latency [4.5.8] - System thread and subprogram models, hierarchical flows. System flows latency less than equal to system latency requirements? Evaluate under synchronous/asynchronous communication.<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify thread states, behavior (thread deadlocks?) | 1. Latency [4.5.8] - Flows through devices, busses, hierarchical flows. System flows latency less than equal to system latency requirements? Evaluate under specified degraded modes of operation.<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify bus modes.<br>3. SSA composed of FMEA, FTA, CCA, RBD, Markov | Comprehensiveness |

| Criteria [AMACC 2021] | Analysis Requirement | Model? | Model as a Black Box | Define Functional Architecture and Modes | Refine to Processes and Threads | Bind to Execution Platforms | Value Type |
|---|---|---|---|---|---|---|---|
| | | | Specify interface type, data, units, behavior | RBD [4.7.7], Markov analysis [4.7.8], SW faults identified, mitigated? 4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent between connected components? | 3. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated. 4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent between connected components? | analysis [4.5.9, 4.5.10], SW faults identified, mitigated 4. Bus, CPU load analysis [4.5.7, 4.6.4, 4.7.4] Determine component degraded properties, analyze wrt capacity requirements 5. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Bus access, behavior/direction consistent between connected components? | |
| | The Vehicle Control Function (VCF) integration with the payload or the control station is evaluated for inability to handle basic faults. | Yes | 1. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify component states, behavior 2. FHA [4.5.10] Identify system-level hazards 3. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Specify interface type, data, units, behavior. | 1.Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Modes consistent? 2. PSSA [4.5.10, 4.5.9] composed of FMEA [4,7,5], FTA [4.7.6], CCA, RBD [4.7.7], Markov analysis [4.7.8], SW faults identified, mitigated? 3. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent between connected components? | 1. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Mode consistent? 2. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated? 3. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent between connected components? | 1. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify Bus modes. Consistently with system mode states? 2. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated? 5. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Bus access, behavior/direction consistent between connected components? | Comprehensiveness |
| | Any synchronization done in any of the integrated functions, including a single synchronization failure or multiple single independent synchronization failures, shall not cause loss of the vehicle/crew and have flying quality levels no worse than Level I. | Yes, but define how to prove "quality flying levels" | 1. Latency [4.5.8] - Model flows, assign latency budgets. 2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify component states, behavior 3. FHA [4.5.10] Identify system-level hazards 4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Specify interface type, data, units, behavior 5. STPA [4.3.2] Define accident, loss of concern, effect of synchronization failure | 1. Latency [4.5.8] - System decomposition, hierarchical flows. Component flows latency <= system latency requirements? Evaluate under synchronous/asynchrono us communication. 2.Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] 3. PSSA [4.5.10, 4.5.9] composed of FMEA [4,7,5], FTA [4.7.6], CCA, RBD [4.7.7], Markov analysis [4.7.8], SW faults identified, mitigated? 4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent | 1. Latency [4.5.8] - System thread and subprogram models, hierarchical flows. System flows latency less than equal to system latency requirements? Evaluate under synchronous/asynchronous communication. 2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify thread states, behavior (thread deadlocks?) 3. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated. 4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior | 1. Latency [4.5.8] - Flows through devices, busses, hierarchical flows. System flows latency less than equal to system latency requirements? Evaluate under synchronous/asynchronous communication. 2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify Processor modes 3. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated? 4. Bus, CPU load analysis [4.5.7, 4.6.4, 4.7.4] Determine component degraded properties, analyze wrt capacity requirements | Risk insight, design insight |

| Criteria [AMACC 2021] | Analysis Requirement | Model? | Model as a Black Box | Define Functional Architecture and Modes | Refine to Processes and Threads | Bind to Execution Platforms | Value Type |
|---|---|---|---|---|---|---|---|
| | | | | between connected components?<br>5. STPA [4.3.2] Iterative development of control charts. Are there unconstrained control actions? Appropriate mitigations? | consistent between connected components?<br>5. STPA [4.3.2] Iterative development of control charts. Are there unconstrained control actions? Appropriate mitigations? | 5. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Bus access, behavior/direction consistent between connected components?<br>5. STPA [4.3.2] Iterative development of control charts. Are there unconstrained control actions? Appropriate mitigations? | |
| | Vehicle control, payload, and ground redundancy of the integrated portions of these functions are able to operate autonomously without loss of the vehicle/crew, and shall not have flying quality levels worse than Level II. | Yes, but define how to prove "quality flying levels" | 1. Latency [4.5.8] - Model flows, assign latency budgets for safe redundancy switch operation<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify component states, behavior related to redundancy switch over<br>3. FHA [4.5.10] Identify system-level hazards<br>4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Specify interface type, data, units, behavior.<br>5. STPA [4.3.2] Define accident, loss of concern | 1. Latency [4.5.8] - System decomposition, hierarchical flows. Component flows latency <= system latency requirements? Evaluate under synchronous/asynchrono us communication.<br>2.Behavioral Consistency [4.5.6, 4.6.3, 4.7.3]<br>3. PSSA [4.5.10, 4.5.9] composed of FMEA [4,7,5], FTA [4.7.6], CCA, RBD [4.7.7], Markov analysis [4.7.8], SW faults identified, mitigated?<br>4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent between connected components?<br>5. STPA [4.3.2] Iterative development of control charts. Are there unconstrained control actions? Appropriate mitigations? | 1. Latency [4.5.8] - System thread and subprogram models, hierarchical flows. System flows latency less than equal to system latency requirements? Evaluate under synchronous/asynchronous communication.<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify thread states, behavior (thread deadlocks?)<br>3. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated.<br>4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent between connected components?<br>5. STPA [4.3.2] Iterative development of control charts. Are there unconstrained control actions? Appropriate mitigations? | 1. Latency [4.5.8] - Flows through devices, busses, hierarchical flows. System flows latency less than equal to system latency requirements? Evaluate under synchronous/asynchronous communication.<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify Processor modes<br>3. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated?<br>4. Bus, CPU load analysis [4.5.7, 4.6.4, 4.7.4] Determine component degraded properties, analyze wrt capacity requirements<br>5. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Bus access, behavior/direction consistent between connected components?<br>5. STPA [4.3.2] Iterative development of control charts. Are there unconstrained control actions? Appropriate mitigations? | Pedagogy, telling the whole story |
| | Verify that vehicle control, payload, and ground system latencies and synchronizations are safe. | Yes, but define "safe operation" | 1. Latency [4.5.8] - Model flows, assign latency budgets<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify | 1. Latency [4.5.8] - System decomposition, hierarchical flows. Component flows latency <= system latency requirements? Evaluate | 1. Latency [4.5.8] - System thread and subprogram models, hierarchical flows. System flows latency less than equal to system latency requirements? | 1. Latency [4.5.8] - Flows through devices, busses, hierarchical flows. System flows latency less than equal to system latency requirements? Evaluate | Comprehensiveness, telling the whole story |

| Criteria [AMACC 2021] | Analysis Requirement | Model? | Model as a Black Box | Define Functional Architecture and Modes | Refine to Processes and Threads | Bind to Execution Platforms | Value Type |
|---|---|---|---|---|---|---|---|
| | | | component states, behavior<br>3. FHA [4.5.10] Identify system-level hazards<br>4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Specify interface type, data, units, behavior.<br>5. STPA [4.3.2] Define accident, loss of concern<br>6. Review modes and mode logic to ensure switching to a fault-tolerant safe state | under synchronous/asynchronous communication.<br>2.Behavioral Consistency [4.5.6, 4.6.3, 4.7.3]<br>3. PSSA [4.5.10, 4.5.9] composed of FMEA [4,7,5], FTA [4.7.6], CCA, RBD [4.7.7], Markov analysis [4.7.8], SW faults identified, mitigated?<br>4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent between connected components?<br>5. STPA [4.3.2] Iterative development of control charts. Are there unconstrained control actions? Appropriate mitigations? | Evaluate under synchronous/asynchronous communication.<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify thread states, behavior (thread deadlocks?)<br>3. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated.<br>4. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Interface type, data, units, behavior consistent between connected components?<br>5. STPA [4.3.2] Iterative development of control charts. Are there unconstrained control actions? Appropriate mitigations? | under synchronous/asynchronous communication.<br>2. Behavioral Consistency [4.5.6, 4.6.3, 4.7.3] Identify Processor modes<br>3. SSA composed of FMEA, FTA, CCA, RBD, Markov analysis [4.5.9, 4.5.10], SW faults identified, mitigated?<br>4. Bus, CPU load analysis [4.5.7, 4.6.4, 4.7.4] Determine component degraded properties, analyze wrt capacity requirements<br>5. Interface Static Consistency Analysis [4.5.5, 4.6.2, 4.7.2] Bus access, behavior/direction consistent between connected components?<br>5. STPA [4.3.2] Iterative development of control charts. Are there unconstrained control actions? Appropriate mitigations? | |
| 9.4.2. The aircraft system shall not display information that degrades safe operation. | An analysis shall be conducted to address the accuracy, integrity, continuity, reliability and related factors to the quality of information provided … to address but not limited to, timing and latency of buses, latency in the presence of single point failures, and latency, BIT error rate and other parameters that define bus integrity and related factors to the quality of data source, the processing of that data, and the subsequent display of information to the crew that could lead to a misleading indication condition... | Yes | RMF [4.5.12] - model CIA impacts and required controls; latency [4.5.8] - model flow budgets; resource loading; [4.5.7] - model message rates, network capacity and error rates | RMF [4.5.12] - refine controls to subsystems; latency [4.6.5] - will subsystems satisfy budgets? | RMF[4.6.12] - are there mixed critical flows within processes?<br>Latency [4.6.5] - will threads satisfy budgets? | RMF [4.7.10] - are controls implemented appropriately?<br>Schedulability [4.7.4] - can we schedule threads to satisfy budgets? Global integrated system scheduling analysis to ensure time lines given fault rates. | Evidence insight, pedagogy |
| | Aircraft/system/subsystem end to end timing and latency shall be tested for normal and fully loaded conditions of all bus components (e.g., | Yes | Latency [4.5.8] - Model flow budgets | Latency [4.6.5] - will subsystems satisfy budgets? | Latency [4.6.5] - will threads satisfy budgets? | Schedulability [4.7.4] - can we schedule threads to satisfy budgets? | Comprehensiveness |

| Criteria [AMACC 2021] | Analysis Requirement | Model? | Model as a Black Box | Define Functional Architecture and Modes | Refine to Processes and Threads | Bind to Execution Platforms | Value Type |
|---|---|---|---|---|---|---|---|
| | networks, switches, hubs) and interfaces for each function. | | | | | | |
| | Bus retries, network data error rates, message size, non-blocking operation, numbers of priorities, and level of compliance with rate monotonic scheduling shall be evaluated. | Yes | Resource loading [4.5.7] - model message rates, network capacity and error rates | Utilization [4.6.4] - will infrastructure handle loads? | Utilization [4.6.4] - will infrastructure handle loads? | Schedulability [4.7.4] - can we schedule threads to satisfy budgets? Formal rate monotonic analysis to evaluate limitations including effects of bus faults, plus integrated or system global processor and bus scheduling analysis. | Design insight |