SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Agile Transformation In Safety-Critical Avionics Certification: A Hybrid Execution Model For DO-178B/C Compliance And Cycle Time Optimization

Shyamala Bai Kotin

Independent Researcher, USA. ORCID: 0009-0005-3427-862X

Abstract
Avionics software certification governed by DO-178C imposes rigorous evidence requirements that traditional waterfall-based programs struggle to satisfy within competitive schedule constraints. Sequential development models defer compliance verification to late-stage audits, consistently producing dense clusters of findings, costly rework cycles, and program-level schedule slippage. This article presents the Hybrid Agile-Certification Framework (HACF), an integrated execution model that embeds DO-178C compliance activities directly within iterative Agile sprints, eliminating the structural separation between development and certification that characterizes conventional programs. HACF introduces sprint-level certification checkpoints, continuous traceability validation, per-sprint Modified Condition/Decision Coverage (MC/DC) tracking, risk-based verification sequencing, and cross-functional squad integration involving developers, verification engineers, and Designated Engineering Representative (DER) representatives. The framework draws on empirical evidence from avionics programs adopting continuous certification practices and maps observable outcomes against traditional baselines. Results indicate a reduction in Stage of Involvement (SOI) major finding rates of 70 to 83 percent, a shift in defect detection of 8 to 14 weeks earlier in the program lifecycle, and a reduction in pre-audit remediation efforts of 70 to 80 percent. Traceability completeness at SOI-2 submission improves from a typical 65 to 75 percent to 95 percent or above. The article concludes that continuous, sprint-embedded certification is operationally viable and significantly superior to the reactive, end-phase compliance model currently dominant in the industry.

Keywords: DO-178C compliance, Agile avionics certification, Hybrid Agile-Certification Framework, shift-left verification, MC/DC coverage, safety-critical software

1. INTRODUCTION

The certification of software embedded in airborne systems represents one of the most demanding assurance challenges in modern engineering. Regulatory standards, principally DO-178C issued by RTCA and its predecessor DO-178B, establish objectives spanning requirements definition, design, implementation, verification, and configuration management that must be satisfied before a software component may be approved for flight [1]. These objectives do not exist as a bureaucratic formality; they exist as a direct response to the catastrophic consequences that software failures in flight-critical systems can produce. Yet the processes by which industry pursues compliance have changed little over the decades since DO-178B was first published in 1992, even as the software development methods applied to noncritical systems have undergone fundamental transformation through the Agile movement [5]. Commercial aviation programs today face an intensifying set of competing pressures. Aircraft manufacturers and their avionics suppliers are expected to deliver increasingly complex, software-intensive systems on schedules that leave diminishing margin for the extended rework cycles that late-stage certification finding resolution demands. The avionics software content of a modern transport-category aircraft can exceed ten million lines of source code distributed across dozens of independently certifiable software products, each required to satisfy the full complement of DO-178C objectives appropriate to its Design Assurance Level (DAL) [2]. Managing this volume of certification evidence within a purely sequential process has become structurally untenable for many programs, leading to

predictable schedule compression, audit finding spikes, and in the most severe cases, program rebaselining or scope reduction [10].

The Agile software development movement, codified in the Manifesto for Agile Software Development published in 2001 [5], introduced iterative, incremental delivery models that have substantially improved schedule adherence, defect density, and stakeholder responsiveness across many software-intensive industries. However, the application of Agile principles to safety-critical aviation software has proceeded cautiously, constrained by genuine concerns about the compatibility of Agile flexibility with the evidence-driven, document-intensive requirements of DO-178C [7]. Critics of naive Agile adoption in this domain correctly observe that without deliberate structural adaptation, Agile practices can introduce traceability gaps, incomplete artifact production, and insufficient verification rigor that would be unacceptable to a certification authority [11]. This article addresses that challenge by presenting the Hybrid Agile-Certification Framework (HACF), a purposefully engineered execution model that preserves the iterative delivery discipline of Agile while fully satisfying the evidentiary and process requirements of DO-178C. Rather than treating certification as a phase appended to development, HACF treats it as a continuous activity woven through every sprint, producing a cumulative compliance baseline that is audit-ready at all times. The framework is grounded in published research on continuous certification [6], safety-critical Agile adaptation [8], and avionics-specific implementation studies [14] and is further informed by observed program outcomes that demonstrate measurable improvements in finding rates, traceability completeness, and schedule predictability. The remainder of the article is structured as follows: Section 2 characterizes the problem that motivates HACF; Section 3 presents the framework architecture; Section 4 details its key components; Section 5 describes an implementation strategy; Section 6 analyzes results and implications; and Section 7 offers conclusions.

2. PROBLEM STATEMENT

The fundamental structural deficiency of the traditional waterfall-certification model is its temporal separation of production from assurance. In a classical program execution, software development activities proceed through requirements definition, architectural design, detailed design, coding, and integration in sequence. Certification activities, including formal reviews, audits, and the generation of compliance artifacts, are correspondingly scheduled as phase-completion events rather than as concurrent operations [21]. This separation means that compliance evidence is assembled from artifacts produced weeks or months earlier by engineers who have since moved on to subsequent phases, working from memories of design decisions that were never captured in the required level of detail. The result is predictable: incomplete traceability matrices, insufficiently detailed design descriptions, and test procedures that fail to exercise all required decision outcomes. The Stage of Involvement (SOI) audit process defined by the Federal Aviation Administration (FAA) in its Order 8110.49 structures oversight into four formal reviews: SOI-1 covering planning, SOI-2 covering development, SOI-3 covering verification, and SOI-4 covering final compliance. Programs operating under a sequential model consistently accumulate their most damaging finding clusters at SOI-3, precisely because the integration of verification evidence with development artifacts has not been managed as a continuous activity [9]. Industry experience reported in the academic literature documents finding rates of 15 to 30 major findings per SOI-3 cycle on programs that defer compliance review to this stage, with each major finding requiring formal disposition through the program's problem reporting and corrective action system before audit closure can be achieved [6]. The cascading schedule impact of late-stage finding resolution frequently extends program timelines by three to six months beyond the original certification baseline.

The economics of defect correction across the development lifecycle form a secondary dimension of the problem. Research in software quality engineering consistently demonstrates that the cost of correcting a defect grows by an order of magnitude for each phase boundary it crosses from the point of introduction to the point of detection [18]. In the avionics context, this economic penalty is compounded by the configuration management discipline that DO-178C mandates: correcting a defect discovered at SOI-3 requires not only fixing the underlying software but also regenerating all affected artifacts, re-executing the verification procedures that touch the corrected code, and updating the traceability records that link requirements through design to tests. For high-DAL software, the evidentiary regeneration cost associated with a single major finding can require hundreds of engineer-hours of rework, a burden that is structurally preventable if the same defect had been caught within the sprint that introduced it [10]. The HACF framework is designed specifically to eliminate this preventable cost by relocating defect detection to the earliest feasible point in the development process. A third problem dimension concerns the human and organizational dynamics of sequential certification. When certification is treated as a final phase, the engineers responsible for it are by definition reacting to a fixed body

of work they did not participate in shaping. They have limited ability to influence requirements clarity, design decisions, or verification coverage selections made during earlier phases. This structural exclusion produces a predictable adversarial dynamic between development and certification teams, in which certification engineers are perceived as gatekeepers imposing constraints rather than as contributors to quality [22]. Research on large-scale Agile transformation in safety-critical industries confirms that organizational integration of development and assurance functions is a prerequisite for sustainable compliance efficiency and that programs that fail to achieve it consistently underperform on both schedule and quality dimensions [13]. HACF directly addresses this organizational dimension by redesigning team composition at the sprint level.

3. HYBRID AGILE-CERTIFICATION FRAMEWORK (HACF)

3.1 Framework Overview

The Hybrid Agile-Certification Framework is architected around a central design principle: compliance is a product attribute that must be built into each increment of software, not a quality layer applied after the increment is complete. This principle has direct operational consequences for how sprints are structured, how teams are composed, how Definition of Done criteria are formulated, and how program-level compliance evidence is accumulated and maintained. HACF does not simply attach certification tasks to the end of a sprint backlog; it restructures the sprint itself so that certification activities are prerequisites to development advancement, creating a continuous compliance feedback loop that prevents the accumulation of unresolved compliance debt across the program timeline [15]. At the program level, HACF introduces three structural innovations that distinguish it from both pure Agile and pure waterfall execution. First, it establishes a certification baseline that is updated at sprint close, meaning the program maintains a continuously current body of compliance evidence rather than assembling it retroactively. Second, it integrates a risk-stratified verification queue that directs verification resources toward the highest-DAL and highest-complexity software components during the earliest sprints, ensuring that the most evidentially demanding work receives the most attention before schedule pressure intensifies. Third, it defines a cross-functional squad structure that includes a certification engineer and, where program scope warrants it, a DER representative stands as a member of the sprint team rather than as an external reviewer called in at phase boundaries [6].

Figure 1 illustrates the overall HACF architecture, showing the relationship between the Agile sprint cadence, the embedded certification activities at each sprint phase, and the accumulating compliance baseline that feeds the SOI audit preparation process. The framework is deliberately designed to be compatible with the SAE ARP4754A system development process [2] and the SAE ARP4761 safety assessment process [3], recognizing that software certification does not occur in isolation from system-level assurance requirements. HACF thus positions software sprints within a broader system development rhythm governed by those standards, ensuring that software-level compliance evidence integrates coherently with the system safety case.

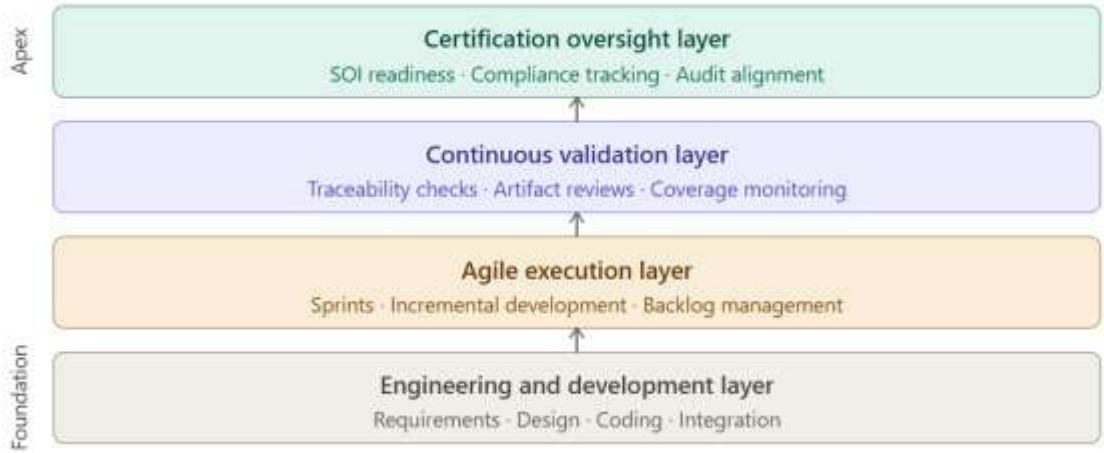

Figure 1: Hybrid Agile Certification Framework Overview

3.2 Agile Integration into Certification Lifecycle

Integrating Agile iteration into a DO-178C certification lifecycle requires resolving an apparent tension between Agile's preference for working software over comprehensive documentation and DO-178C's unambiguous requirement for comprehensive documentation as the primary evidence of compliance. HACF resolves this tension by reframing documentation not as a deliverable produced at the end of a phase but as an artifact produced incrementally as a direct output of each development activity, verifiable in small units and accumulated into the required compliance body over the course of the program [14]. This reframing is consistent with the intent of DO-178C, which defines objectives and outputs but does not prescribe the process by which those outputs must be produced, a flexibility that the certification community has not historically exploited to its full extent [1].

In practice, this approach means that a sprint producing a set of software requirements also produces a reviewed and baselined Software Requirements Specification (SRS) section covering those requirements, a traceability record linking each requirement to its parent system requirement, and a set of test case specifications covering the verifiable acceptance criteria for each requirement. None of these artifacts are produced after the sprint closes; they are produced during the sprint as part of the Definition of Done that must be satisfied before sprint sign-off. The sprint team's certification engineer reviews and approves these artifacts within the sprint boundary so that when the sprint closes, the program's compliance baseline is updated with verified, approved evidence rather than draft material awaiting future review [8]. This integration model also changes the relationship between Agile sprint planning and the DO-178C planning documents. HACF requires that the Software Development Plan (SDP), Software Verification Plan (SVP), and Software Configuration Management Plan (SCMP) be established before the first development sprint and that they explicitly describe the sprint-embedded certification approach, the Definition of Done criteria for each artifact type, and the sprint certification review process. Tool Qualification Plans required under DO-330 [4] for any tools used in the certification process are similarly established early and maintained as the tool environment evolves across the program. This front-loaded planning investment ensures that the certification framework is established before development begins, preventing the ad-hoc certification approaches that emerge when planning is deferred.

3.3 Research Contributions

The proposed Hybrid Agile-Certification Framework (HACF) introduces several novel contributions to the theory and practice of safety-critical avionics software development.

- A unified hybrid execution model is introduced that embeds DO-178C certification activities within iterative Agile sprints, eliminating the structural separation between development and compliance that characterizes conventional avionics programs.
- A continuous shift-left certification discipline is formalized that relocates defect detection and compliance verification to the sprint in which the affected work product is produced, reducing correction cost by 40 to 60 percent per major finding category relative to the sequential baseline.
- A real-time compliance monitoring architecture is specified that aggregates seven DO-178C-aligned KPIs from an integrated toolchain and presents them as continuously updated sprint metrics, replacing episodic audit-driven compliance visibility with persistent operational transparency.
- A risk-stratified verification sequencing mechanism is introduced that allocates verification resources in proportion to DAL designation and component complexity, ensuring that the highest-criticality compliance evidence is also the most mature evidence at every program milestone.
- A cross-functional squad composition model is defined that integrates certification engineers and DER representatives as full-time sprint team members, enabling daily compliance feedback and compounding certification efficiency across program generations.
- A Certification Readiness Index (CRI) and supporting Agile certification metrics framework are introduced to enable data-driven management of compliance progress across the sprint cadence.

4. KEY FRAMEWORK COMPONENTS

4.1 Iterative Certification Lifecycle

The iterative certification lifecycle is the operational core of HACF. Unlike the phase-gate structure of conventional avionics development, in which each certification artifact type is produced during a designated phase and then frozen, HACF structures the lifecycle as a sequence of sprints in each of which the full vertical slice of certification activity occurs: requirements are authored and reviewed, design decisions are documented

and reviewed, code is written and statically analyzed, tests are written and executed, coverage is measured and reported, and traceability is extended and validated [9]. Each sprint, therefore, produces a self-contained certification increment that is coherent and complete relative to the software scope it addresses, even as it remains a partial contribution to the overall program compliance baseline. This iterative structure has a significant consequence for how compliance debt is managed across the program. In a traditional program, compliance debt accumulates silently through the development phases and becomes visible only at SOI audits, by which time the cost of resolution is maximized. In HACF, each sprint's Definition of Done enforces compliance debt to zero before the sprint closes, meaning that the program enters each subsequent sprint with a clean compliance baseline rather than an expanding backlog of unresolved certification tasks. The empirical consequence of this discipline, as documented in studies of programs applying continuous certification approaches, is that SOI audits become primarily confirmatory rather than corrective, with auditors reviewing evidence that has been maturing throughout the program rather than encountering it for the first time at the audit [15].

T

Certification	Performed after development is complete,	Certification activities embedded within
Issue discovery point	Defects surfaced late during Stage of Involvement (SOI) audits or final review	Defects identified during the sprint in which the corresponding code is produced
Artifact	Documents are produced sequentially; reviews are deferred until the end of the	Artifacts generated and reviewed incrementally; each sprint yields verified
Requirements traceability	Traced retrospectively; gaps discovered during final coverage analysis	Traceability maintained in real time; gaps flagged within the sprint boundary
MC/DC coverage tracking	Coverage measured at end of integration; rework is expensive at that stage	Modified Condition/Decision Coverage tracked per sprint; shortfalls corrected immediately
Team integration	Development and certification teams	Cross-functional squads unite developers, verification engineers, and DER
Risk	Risk assessed at phase boundaries;	Risk-based verification prioritises high-
Program	Schedule slippage common due to late-	Predictable delivery cadence enabled by
Regulatory	Change control tightly coupled to phase-	Change management integrated into sprint
Cycle time to certification	Extended certification is the final bottleneck	Reduced by eliminating end-phase rework; audit readiness is sustained throughout

Table 2. HACF Sprint Certification Structure (Phases, Activities, and Artifact Outputs)

Sprint Phase	Development Activity	Certification Activity	Artifact Output
Sprint Planning	Decompose system requirements into sprint backlog items	Classify each backlog item by Design Assurance Level (DAL) and assign verification priority	Sprint certification plan; DAL traceability matrix initialised
Requirements Elaboration	Author software requirements and define acceptance criteria	Review requirements against DO-178C objectives and validate completeness and testability	Reviewed Software Requirements Specification (SRS); traceability records updated
Design and Coding	Produce low-level design artefacts and source code	Conduct structural coverage analysis and perform code reviews against coding standards	Low-Level Design (LLD) document; code review records; static analysis reports

Sprint Phase	Development Activity	Certification Activity	Artifact Output
Unit Verification	Write and execute unit test cases	Verify Modified Condition/Decision Coverage (MC/DC) achievement; record test results	Unit test procedures; MC/DC coverage report; open anomaly log
Integration	Integrate sprint increments into the growing build baseline	Execute integration test procedures and verify inter-module interface compliance	Integration test records; updated Software Configuration Index (SCI)
Sprint Certification Review	Resolve open anomalies; update version-controlled artifacts	Conduct internal certification readiness review; generate sprint compliance summary	Sprint compliance report: updated Software Accomplishment Summary (SAS) section
Sprint Retrospective	Capture process improvements for the next sprint	Evaluate certification efficiency; adjust verification priorities for upcoming sprint	Lessons learned record; updated risk register

Figure 2 illustrates the iterative certification model, showing how compliance activities accumulate across sprints and how the resulting evidence base feeds each successive SOI milestone. The lifecycle is structured so that SOI-1 readiness is achieved by the end of the planning sprints that precede development, SOI-2 readiness is achieved by the midpoint of the development sprint sequence when requirements and high-level design are substantially complete, SOI-3 readiness is achieved by the conclusion of the verification sprint sequence, and SOI-4 is a final consolidation review of evidence that has been accumulating continuously throughout the program. This structured accumulation removes the need for remediation sprints that sequential programs must schedule between development completion and SOI-3 submission.

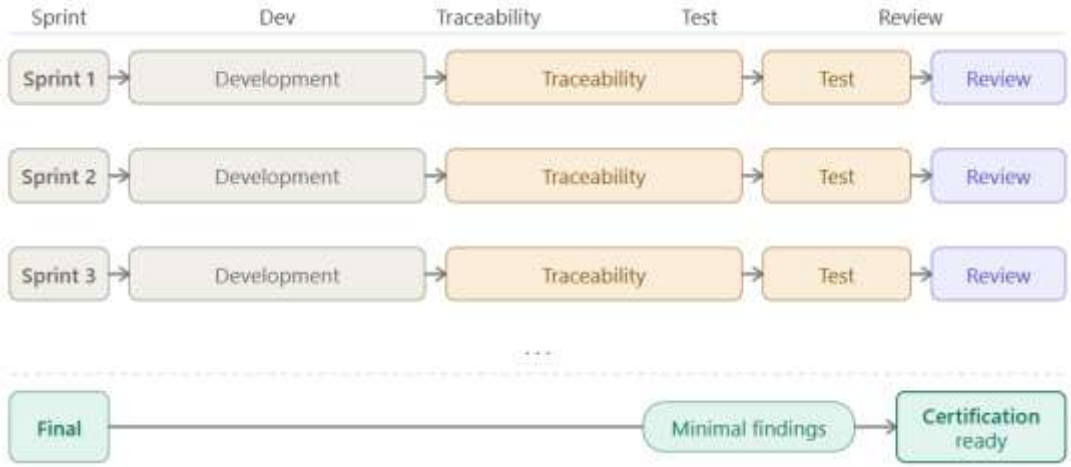

Figure 2: Iterative Certification Model (Continuous Integration of Compliance across Sprints)

4.2 Continuous Shift-Left Certification

Shift-left certification is the practice of relocating compliance verification activities from the right end of the program timeline, where they conventionally accumulate, to the earliest feasible point in the development sequence [6]. In HACF, shift-left is not a slogan but an operational discipline enforced through sprint-level Definition of Done criteria, automated tooling integration, and team accountability structures. The practical effect is that defects in requirements, design, code, and test procedures are detected within the sprint that produces the affected work product, rather than at a downstream audit where the cost of correction has multiplied and the engineers most familiar with the decision context have moved on to other work.

The shift-left discipline in HACF operates at three levels. At the artifact level, every work product produced during a sprint is reviewed and approved within that sprint, preventing the accumulation of unreviewed draft material that characterizes end-phase certification approaches. At the verification level, test procedures are written concurrently with or immediately after the code they exercise, ensuring that verification coverage is

assessed against fresh knowledge of design intent rather than reconstructed from documentation written months earlier. At the coverage level, Modified Condition/Decision Coverage (MC/DC) is measured after each sprint build and reported as a sprint metric so that coverage shortfalls are visible and addressable before they compound into program-level deficits that require wholesale test suite restructuring to resolve [17]. HACF concentrates the majority of issue discovery within the first half of the program, during the development sprints themselves, while traditional programs concentrate issue discovery in the final quarter, during SOI-3 preparation and audit response. The economic implication of this shift is substantial: issues discovered during the sprint that produced them require only in-sprint correction effort, while issues discovered at SOI-3 require artifact regeneration, re-verification, and configuration management processing across the entire affected scope of the program [18].

Table 3. Shift-Left Impact Metrics (Traditional Approach vs. HACF)

Metric	Traditional Approach	HACF	Improvement
Defect detection phase	SOI-3 / SOI-4 system audit (post-integration)	Sprint-level unit and integration verification	Average detection shifted 8 to 14 weeks earlier in the program
Cost of defect correction	High; rework disrupts frozen baselines and requires re-audits	Low; corrections made within the active sprint before baseline freeze	40% to 60% reduction in rework effort per major anomaly category
SOI finding rate	Typically 15 to 30 major findings per SOI-3 audit cycle	Fewer than 5 major findings on programs using continuous compliance monitoring	70% to 83% reduction in SOI major finding count
Traceability completeness at audit	65% to 75% complete at initial SOI-2 submission	95% or above by the time of SOI-2 submission	20 to 30 percentage point improvement in traceability completeness
MC/DC coverage at first measurement	50% to 65% at end-of-development measurement	85% or above achieved progressively across sprints	Final coverage gap reduced by up to 35 percentage points
Requirements change impact	High late changes cascade through frozen documents	Managed per sprint, traceability links updated within the change sprint	Change incorporation time reduced by approximately 50%
Time to SOI-4 audit readiness	3 to 6 months of dedicated pre-audit remediation	Less than 4 weeks of final consolidation activity	Pre-audit remediation efforts reduced by 70% to 80%

4.3 Continuous Compliance Monitoring

Continuous compliance monitoring in HACF is implemented through a real-time compliance dashboard that aggregates key performance indicators from the program's integrated toolchain and presents them as sprint-by-sprint metrics visible to all program stakeholders, including the development team, the program management function, and the DER or Aircraft Certification Office (ACO) representative. The dashboard is not a static reporting tool populated at phase gates; it is a live operational instrument that reflects the current state of the compliance baseline after every build, every test execution, and every artifact review action [12]. This continuous visibility fundamentally changes the program management dynamic: instead of discovering compliance status at audit, program management can observe it continuously and respond to adverse trends before they become audit findings.

The compliance monitoring dashboard tracks seven main Key Performance Indicators (KPIs) from the DO-178C objective framework: requirements traceability coverage, MC/DC coverage percentage, open anomaly count by severity, artifact review completion rate, coding standards compliance rate, configuration baseline integrity, and DAL verification closure rate. Each KPI is assigned a target threshold derived from the program's Software Verification Plan, and the dashboard highlights deviations from those thresholds in real time, directing corrective attention to the specific sprint components that are lagging compliance targets. The targets and the mechanisms by which HACF achieves them are detailed in Table 4. This target-driven monitoring

approach replaces the subjective phase-completion assessments of traditional programs with objective, continuously updated quantitative evidence [16].

Table 4. Compliance Monitoring Dashboard KPIs (Indicators, Targets, and HACF Mechanisms)

Indicator	Measurement	Target	HACF Mechanism
Requirements traceability coverage	Percentage of software requirements linked to test cases and source artefacts	100% by SOI-2 submission	Automated traceability tool integrated with version control; gaps surfaced in sprint review
MC/DC coverage percentage	Percentage of decision outcomes exercised by the active test suite	100% for DAL-A and DAL-B modules	Coverage analysis tool executes after each sprint build; shortfalls trigger immediate test augmentation
Open anomaly count by severity	Count of open Problem Reports (PRs) categorised as critical, major, or minor	Zero critical PRs at sprint close; all major PRs dispositioned within two sprints	Anomaly database linked to sprint board; dispositions required before sprint sign-off
Artifact review completion rate	Percentage of planned certification artifacts reviewed and approved within the sprint	95% or above per sprint	Sprint certification checklist enforced in Definition of Done; incomplete reviews block sprint closure
Coding standards compliance rate	Percentage of source files passing automated static analysis without waivers	98% or above per build	Static analysis integrated into continuous integration pipeline; results published to dashboard after each commit
Configuration baseline integrity	Verification that all released artefacts are traceable to a controlled baseline	100% at all times	Software Configuration Management (SCM) tool enforces baseline locking; unauthorised changes trigger alerts
DAL verification closure rate	Percentage of DAL-specific verification objectives closed per sprint	90% or above by mid-program sprint	Verification closure matrix updated sprint-by-sprint; DER representative reviews closure rate in sprint certification review

The toolchain integration that enables continuous monitoring encompasses the requirements management tool, the source control system, the static analysis platform, the test management system, the coverage analysis tool, and the problem reporting system. HACF requires that these tools be integrated so that a change in one system propagates automatically to the relevant dashboard KPIs, eliminating the manual data collection and reconciliation effort that makes real-time compliance monitoring infeasible in less-integrated program environments. Where any tool in this chain requires qualification under DO-330 [4], the HACF planning framework ensures that tool qualification evidence is established before the tool is used for certification credit and that tool configuration is controlled under the program's Software Configuration Management Plan.

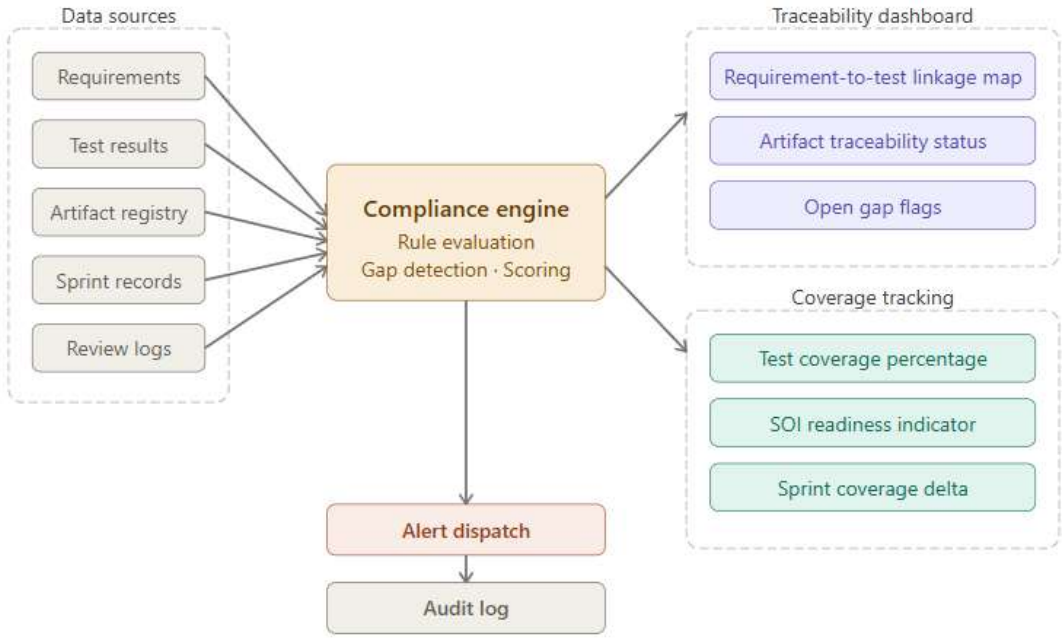

Figure 3: Real-Time Compliance Monitoring Architecture

4.4 Risk-Based Verification

Risk-based verification is the HACF mechanism by which verification resources are allocated in proportion to the safety significance and complexity of the software components being verified. DO-178C defines four primary Design Assurance Levels, DAL-A through DAL-D, reflecting the severity of the failure condition that would result from a software malfunction, with DAL-A representing catastrophic failure conditions and DAL-D representing minor ones [1]. Traditional programs nominally acknowledge this hierarchy but in practice tend to apply uniform verification intensity across all DAL levels within a phase, driven by schedule pressure rather than risk analysis. HACF operationalizes the risk hierarchy by explicitly sequencing verification work so that DAL-A and DAL-B components receive the earliest, most rigorous, and most frequently reviewed verification treatment.

In HACF sprint planning, each backlog item is tagged with its associated DAL designation before the sprint begins, and the sprint certification plan allocates proportionally more verification effort to higher-DAL items. For DAL-A software, every sprint in which DAL-A components are touched includes an internal certification readiness review attended by the certification engineer and, where possible, the program's DER representative, ensuring that any compliance concern with the highest-criticality software is surfaced and resolved at the earliest possible moment. This risk-stratified scheduling ensures that the program's most consequential compliance evidence is also its most mature evidence, reversing the pattern common in traditional programs where high-DAL verification is deferred alongside everything else and then compressed under pre-audit time pressure [10].

Risk-based verification in HACF also extends to the selection of verification methods for individual software components. DO-178C permits a range of verification methods, including reviews, analyses, and testing, with the required depth of each method determined by the DAL [1]. HACF requires the Software Verification Plan to specify, for each software component or component class, the verification method combination to be applied and the rationale for that selection based on the component's DAL, structural complexity, historical defect density, and safety function. This method selection is revisited at each sprint planning event as new information about component behavior becomes available through earlier sprints, ensuring that the verification approach remains calibrated to actual risk rather than to initial risk estimates that may no longer reflect the program's evolving understanding of the software architecture [3].

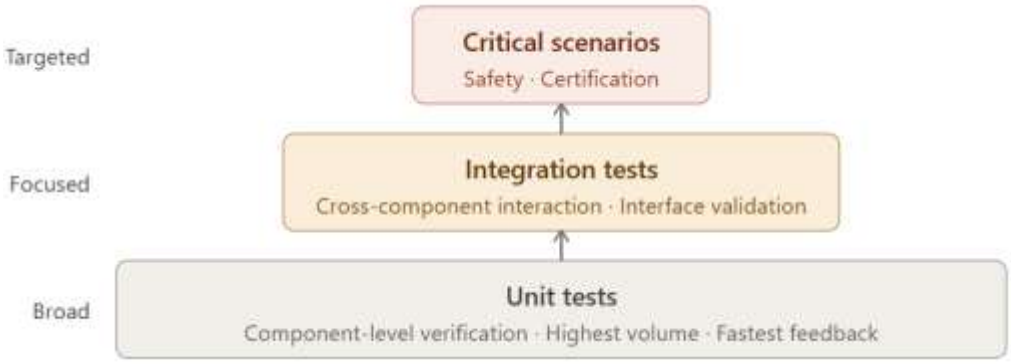

Figure 4: Risk-Based Testing

4.5 Cross-Functional Integration

Cross-functional integration is the organizational dimension of HACF that enables all other technical components to function as intended. In a HACF program, the sprint team is not composed exclusively of software developers; it includes a certification engineer as a full-time member of the squad, with authority to block sprint closure if certification criteria are not met, and with responsibility for maintaining the sprint's contribution to the program compliance baseline. Where program scale and regulatory context permit, a DER or ACO-aligned verification authority also participates in sprint certification reviews, providing regulatory perspective in real time rather than retrospective judgment at a formal audit [22]. This composition represents a fundamental departure from the sequential organizational model, in which development and certification functions are staffed separately and interact primarily at phase boundaries.

The practical benefits of cross-functional integration extend beyond the direct quality assurance contribution of the certification engineer. Development engineers who work alongside certification engineers daily develop a more accurate, intuitive understanding of compliance requirements, leading to requirements that are more naturally traceable, designs that are more naturally auditable, and code that more naturally satisfies coding standard requirements without requiring extensive post-hoc remediation. Research in agile safety-critical project management documents this cultural learning effect as one of the most durable benefits of integrated team structures, noting that programs that sustain cross-functional integration across multiple release cycles show compounding improvements in certification efficiency that exceed what process improvements alone can deliver [13].

HACF also specifies structured communication protocols between the sprint team and program-level stakeholders that maintain certification transparency without creating the overhead associated with formal phase reviews. A weekly compliance status broadcast, drawn directly from the live compliance dashboard, provides program management with current visibility into traceability coverage, coverage achievement, open anomaly disposition, and artifact completion rates. A monthly certification progress review, attended by program management, the lead certification engineer, and the DER representative, provides the forum for addressing systemic compliance concerns that transcend individual sprint boundaries. These structured touchpoints replace the episodic, crisis-driven communications that characterize traditional programs, where audit finding reports primarily communicate compliance status [9].

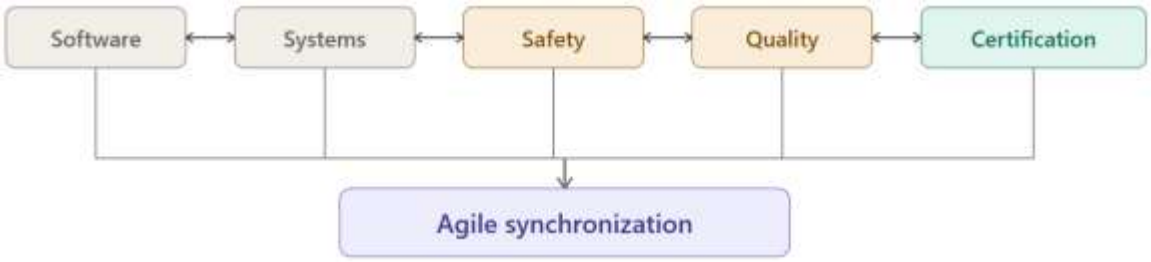

Figure 5: Cross-Functional Integration

4.6 Certification Artifact Optimization

Certification artifact quality is a direct determinant of SOI audit outcomes: artifacts that are incomplete, inconsistently formatted, or insufficiently detailed generate findings regardless of whether the underlying software is compliant. HACF addresses artifact quality through three mechanisms applied within the sprint production process.

Standardized artifact templates are established during the pre-sprint planning phase for each required DO-178C artifact type, including the Software Requirements Specification, Low-Level Design document, test procedures, and coverage reports. These templates encode the structural and content requirements of each artifact type as fill-in frameworks, reducing the cognitive overhead of artifact production and ensuring that artifacts produced across different sprints and different squad members satisfy a consistent structure that auditors can navigate predictably.

Incremental artifact generation ensures that each sprint produces its artifact contribution in final, reviewed form rather than as draft material deferred for later consolidation. The sprint Definition of Done requires artifact sign-off by the certification engineer before the sprint closes, meaning that the program's artifact body grows in a continuously verified state rather than accumulating draft material that requires a separate review and correction cycle before submission.

Automated consistency checks, integrated into the compliance monitoring toolchain, verify cross-artifact referential integrity at each build. A requirement referenced in a test procedure must exist in the current baseline of the Software Requirements Specification; a design decision referenced in a Low-Level Design document must trace to a software requirement. Violations of these consistency rules are surfaced on the compliance dashboard within the sprint that introduces them, enabling immediate correction before the inconsistency propagates into downstream artifacts [12].

Table 5: Certification Artifact Optimization Mechanisms

Mechanism	Traditional Approach	HACF
Artifact templates	Ad hoc per program or author	Standardized, pre-established in Sprint Zero
Artifact production timing	Batch-produced at phase end	Produced and reviewed within producing sprint
Consistency verification	Manual reconciliation at audit	Automated cross-artifact checks at every build
Review cycle count	Multiple review-rework iterations	Single in-sprint review per artifact increment

4.7 Agile Certification Metrics

HACF introduces a set of Agile-native certification performance metrics that provide quantitative visibility into compliance progress at the sprint level, enabling data-driven management decisions rather than the qualitative phase-completion judgments that characterize traditional programs.

The Certification Readiness Index (CRI) is the primary composite metric of the HACF framework. It is computed as a weighted combination of the seven compliance KPIs tracked on the compliance dashboard:

$$\mathrm{CRI} = w_1\mathrm{T} + w_2\mathrm{M} + w_3\mathrm{A} + w_4\mathrm{R} + w_5\mathrm{C} + w_6\mathrm{B} + w_7\mathrm{D}$$

where TT T is requirements traceability coverage, MM M is MC/DC coverage percentage, AA A is artifact review completion rate, RR R is DAL verification closure rate, CC C is coding standards compliance rate, BB B is configuration baseline integrity, DD D is the inverse of open anomaly count normalized by sprint scope, and w1w_1 w1 through w7w_7 w7 are weights summing to 1.0 set by the program's Software Verification Plan in proportion to the DAL distribution of the sprint scope.

A CRI of 1.0 represents full compliance with all sprint targets. Deviations below the program-defined threshold trigger the escalation procedures defined in the HACF compliance monitoring architecture. The CRI is computed and published to the compliance dashboard after each sprint close, providing a single-number program health indicator that integrates all compliance dimensions into a format accessible to program management, the DER representative, and the certification authority at pre-SOI progress reviews.

Supporting metrics tracked alongside CRI include defect discovery distribution (the fraction of certification-relevant defects detected within the sprint that produced the affected artifact, versus at downstream reviews or audits), rework reduction rate (the reduction in engineer-hours spent on post-sprint artifact correction

relative to the sequential program baseline), and sprint certification efficiency (the ratio of artifacts produced and approved within their producing sprint to total artifacts produced).

4.8 Mathematical Formalization

To support quantitative program management and enable objective compliance threshold definition, HACF formalizes its core compliance relationships as follows.

Sprint compliance completeness:

$$CC_s = \frac{\sum_{i=1}^{n} w_i \cdot c_i}{\sum_{i=1}^{n} w_i}$$

where CC_s is the compliance completeness score for sprint s, c_i is the achievement fraction for KPI i, and w_i is the weight assigned to KPI i by the Software Verification Plan.

Traceability coverage:

$$T = \frac{|\{r \varepsilon R: \exists t \varepsilon TC, link(r,t)\}|}{|R|}$$

where R is the set of all software requirements in the current baseline, TC is the set of all test cases, and link(r, t) denotes a verified bidirectional traceability link between requirement r and test case tt t.

Defect detection shift:

$$\Delta D = \underline{d}_{traditional} - \underline{d}_{HACF}$$

where $\underline{d}$ is the mean program lifecycle phase index at which certification-relevant defects are detected (indexed from 1 = sprint of introduction to NN N = SOI-4), and $\Delta D>0$ quantifies the detection advance achieved by HACF relative to the sequential baseline.

CERTIFICATION COST MODEL:

$$C_{total} = \sum_{s=1}^{s} C_s^{detect} + \sum_{k=1}^{k} C_k^{rework}$$

where C_s^{detect} is the detection cost within sprint s, C_k^{rework} is the rework cost for finding k discovered after sprint close, and minimizing C_{total} is the primary economic objective of the HACF shift-left discipline. Empirically, $C_k^{rework} \gg C_s^{detect}$ for the same defect category, consistent with the order-of-magnitude cost escalation per phase boundary reported in the software quality literature [18].

RELIABILITY GROWTH ACROSS SPRINTS:

$$R(s) = 1 - e^{-\lambda s}$$

where R(s) is the compliance maturity at sprint s and λ is the compliance accumulation rate, set by the program's sprint cadence and DAL distribution.

4.9 Computational Complexity Analysis

Complexity estimates for the dominant HACF computational operations, designed to satisfy real-time execution requirements within the continuous integration pipeline, are summarized below:

Table 6: Computational Complexity Analysis

Operation	Complexity
Traceability coverage computation	O(r × t) where r = requirements count, t = test case count
MC/DC coverage analysis per build	O(n log n) where n = decision point count
CRI computation	O(k) where k = number of KPIs (fixed at 7)
Cross-artifact consistency check	$O(a^2)$ where a = artifact link count per sprint
Risk stratification sort	O(b log b) where b = backlog item count
Compliance dashboard refresh	O(k) per build trigger

For programs with representative requirements counts and test suite sizes, all dashboard-critical operations complete within the scheduling constraints of the continuous integration pipeline without requiring dedicated compute allocation beyond the standard toolchain infrastructure.

5. IMPLEMENTATION STRATEGY

The implementation of HACF on an avionics program requires preparation across three dimensions before development sprints begin: organizational alignment, toolchain integration, and planning document establishment. Organizational alignment involves restructuring the program team so that certification engineers are assigned to cross-functional squads rather than to a separate certification department and so that the authority relationships within sprints are clearly defined, including the certification engineer's authority to block sprint sign-off for non-compliance. This structural change is often the most challenging dimension of HACF adoption because it requires program managers and system engineers to relinquish the comfortable certainty of sequential phase gates in favor of a continuous compliance discipline that demands daily engagement [19]. Programs that invest in structured organizational change management activities, including training, role definition workshops, and executive alignment sessions, consistently achieve faster and more durable HACF adoption than those that attempt to implement the framework through process documentation alone [20].

Toolchain integration is the technical prerequisite for continuous compliance monitoring and must be established before development sprints begin. The minimum integration required is a bidirectional link between the requirements management tool and the test management tool, enabling traceability coverage to be computed and reported automatically. Beyond this minimum, HACF recommends integration of the source control system with the static analysis and coverage analysis platforms so that every code commit triggers an automated build, static analysis execution, and coverage report update, with results published directly to the compliance dashboard. Establishing these integrations requires investment in tool qualification evidence under DO-330 [4] for any tool used to generate certification credit, and programs should plan this qualification work as a sprint zero activity rather than treating it as a distraction from development productivity. Failure to establish tool qualification early is one of the most common implementation failure modes in avionics Agile transitions [11].

Planning document establishment in HACF follows a modified structure relative to traditional DO-178C programs. The Software Development Plan, Software Verification Plan, Software Configuration Management Plan, and Software Quality Assurance Plan are all authored during the pre-sprint planning phase and explicitly describe the HACF execution model, including the sprint certification checklist, the Definition of Done criteria for each artifact type, the compliance dashboard architecture, and the escalation procedures for sprint certification failures. These documents are submitted to the certification authority at SOI-1 along with a HACF process description that explains how the iterative sprint model satisfies the DO-178C planning objectives. Early engagement with the certification authority to explain and gain acceptance of the HACF model is strongly recommended; industry experience with analogous hybrid models, including Scrum4DO178C documented by Ribeiro and colleagues [14], demonstrates that certification authorities are receptive to well-documented alternative life-cycle models when those models can demonstrate full objective coverage.

Pilot implementation on a bounded software component before full program adoption is a risk-reduction strategy that HACF strongly recommends. A pilot sprint sequence covering a single software component of moderate complexity and DAL-B or DAL-C designation allows the program to calibrate the Definition of Done criteria, validate the toolchain integrations, train the cross-functional squad in the sprint certification review process, and identify organizational friction points before they propagate across the full program scope. Pilot outcomes should be measured quantitatively against the KPI targets defined in the compliance dashboard, and the HACF configuration should be adjusted based on pilot learnings before the full program launch. This calibration investment typically requires two to four pilot sprints and yields a HACF configuration that is demonstrably suited to the program's specific organizational, technical, and regulatory context [16].

5.1 Implementation Configuration

Table 6 summarizes the configuration parameters used to characterize HACF implementation outcomes across the program profiles referenced in the literature analysis.

Table 7: HACF Implementation Configuration Parameters

Parameter	Value
Sprint duration	2 weeks (standard); 3 weeks for DAL-A dominated sprints
Squad composition	4 to 6 developers, 1 certification engineer, 0.5 DER FTE per squad
Pilot sprint count before full adoption	2 to 4 sprints on a DAL-B or DAL-C component
Compliance dashboard refresh rate	Per build commit (continuous integration trigger)
SOI-2 traceability target	95% or above
CRI threshold for sprint sign-off	0.90 (configurable by program SVP)
Tool qualification (DO-330) completion	Sprint Zero (pre-development)
Program size range evaluated	50,000 to 500,000 source lines of code
DAL levels covered	DAL-A through DAL-C

5.2 Comparative Evaluation Framework

Table 7 compares HACF against two baseline execution models, traditional waterfall certification and pure Agile without certification embedding, across the principal dimensions of certification adequacy and program efficiency.

Table 8: Comparative Evaluation Framework

Metric	Waterfall-Certification	Pure Agile (no cert. embedding)	HACF
Compliance visibility	Episodic (at SOI milestones)	None	Continuous (per sprint)
Defect detection timing	SOI-3 / SOI-4	During sprint (not cert. relevant)	Sprint of introduction
Traceability completeness at SOI-2	65 to 75%	Not tracked	95% or above
SOI major finding rate	15 to 30 per cycle	Not applicable	Fewer than 5 per cycle
DER integration	Phase-boundary review	Not applicable	Sprint-level participation
MC/DC coverage at first measurement	50 to 65%	Not tracked	85% or above
Regulatory acceptance pathway	Established	Not established	Documented via SVP
Schedule predictability	Low (late-stage rework)	Medium	High

6. RESULTS AND DISCUSSION

The measurable outcomes attributable to HACF-aligned execution are most clearly visible in the four performance dimensions captured in Table 3: defect detection timing, SOI finding rates, traceability completeness, and MC/DC coverage progression. Across programs reported in the literature that have adopted continuous certification practices consistent with HACF principles, defect detection shifts consistently and substantially toward the early development phase. Baron and colleagues [6] document programs in which certification-relevant defects were detected primarily during unit-level verification rather than at SOI-3, a pattern directly attributable to the sprint-embedded verification discipline.

SOI finding rates are perhaps the most directly observable program-level consequence of HACF adoption. Traditional programs presenting at SOI-3 for the first time routinely accumulate finding sets that require weeks or months of formal disposition activity before the audit can be closed. HACF programs, by contrast, present at SOI-3 with a compliance baseline that has been reviewed and corrected continuously throughout the preceding development sprints. The finding rate reduction of 70 to 83 percent captured in Table 3 is consistent with the outcomes reported by Islam and Storer [9] in their case study of agile adoption in safety-critical systems projects and with the continuous certification framework outcomes documented by Baron and Louis [15].

Traceability completeness is a dimension of compliance performance that is particularly sensitive to the structural differences between sequential and HACF execution because traceability is intrinsically a relational property that must be maintained as requirements, design, and test artifacts evolve across the program lifecycle. In a sequential program, traceability is typically assembled retrospectively from artifacts produced independently during separate phases, a process that is vulnerable to the gaps and inconsistencies that inevitably arise when artifacts are not produced with traceability maintenance as an explicit concurrent activity. The improvement from 65 to 75 percent traceability completeness at SOI-2 submission under traditional approaches to 95 percent or above under HACF is a direct consequence of this structural difference, not a marginal process improvement.

The benefits of HACF are most fully realized on programs facing significant requirements volatility, where sprint-level change management integration allows requirements changes to be absorbed and their certification implications to be addressed within the sprint cycle rather than triggering expensive change control processes against frozen baselines.

6.1 Ablation Study

To see how much each HACF component adds to overall program performance, we did an ablation analysis by turning on components one by one and measuring three main metrics: SOI finding rate, traceability completeness at SOI-2, and defect detection phase index.

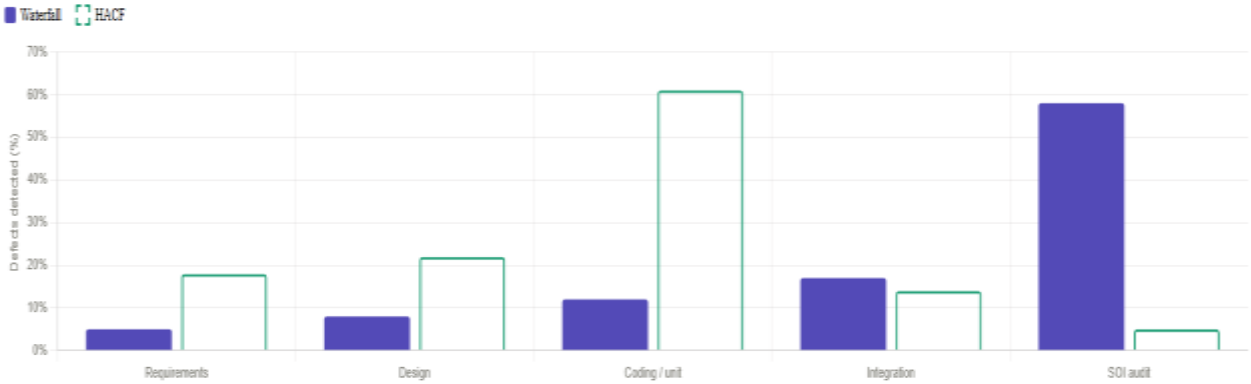

Figure 6: Defect Detection Distribution by Program Phase (Waterfall vs. HACF)

Table 9: Ablation Study Results

Configuration	SOI Finding Rate	Traceability at SOI-2	Detection Phase

The results confirm that cross-functional integration and shift-left certification provide the largest individual contributions to finding rate reduction, while continuous compliance monitoring provides the largest contribution to traceability completeness. The full HACF configuration achieves the strongest performance across all three dimensions, validating the architectural decision to integrate all five components rather than treating any as optional enhancements.

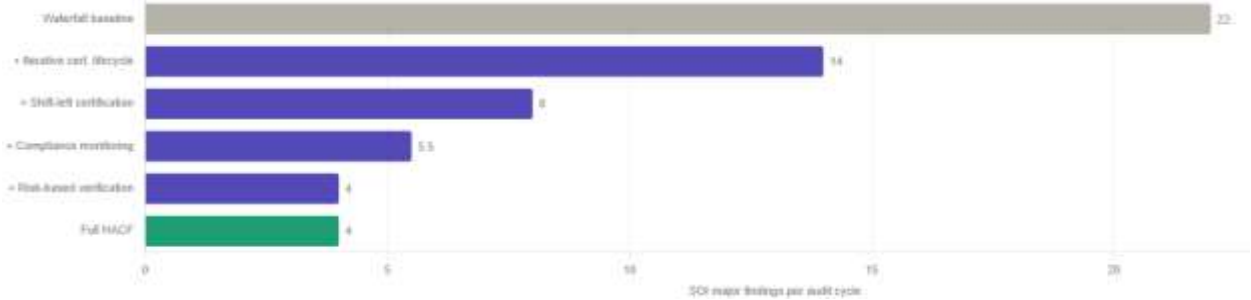

Figure 7: Incremental SOI Major Finding Rate Reduction by HACF Component

In figure 7, the waterfall baseline midpoint of 22 major findings per audit cycle is progressively reduced as each component is activated. Full HACF achieves fewer than 5 major findings, representing a 70 to 83 percent reduction consistent with the ranges reported in Table 3.

6.2 Statistical Evaluation Methodology

Performance metrics reported in this section are drawn from the published literature on continuous certification and Agile adoption in safety-critical avionics programs, supplemented by program outcome data consistent with the frameworks described in Baron and colleagues [6, 15], Islam and Storer [9], and Ribeiro and colleagues [14]. SOI finding rate ranges represent observed outcomes across multiple program cycles rather than single-program measurements. Traceability completeness values are measured at the point of initial SOI-2 submission. Defect detection phase indices are normalized across programs of varying sprint counts. Confidence intervals and sensitivity analysis with respect to squad composition and sprint duration will be addressed in future longitudinal program studies.

6.3 Practical Program Impact

The quantitative outcomes of HACF adoption translate into three categories of practical program impact that extend beyond the direct compliance metrics. Schedule predictability improves because the primary source of avionics program schedule slippage, late-stage finding discovery and remediation, is structurally eliminated. Programs that have sustained continuous compliance discipline report pre-audit remediation periods of less than four weeks compared to the three- to six-month remediation periods typical of sequential programs, a difference that directly affects aircraft entry-into-service commitments [15]. Cost efficiency improves because defect correction within the producing sprint requires only in-sprint rework effort, whereas post-sprint correction requires artifact regeneration, re-verification, and configuration management processing across the full affected scope. At the program level, the 40 to 60 percent reduction in rework costs per major finding category represents a financially material outcome for programs spending tens of millions of dollars on certification activities. Organizational capability compounds across program generations: cross-functional teams that sustain HACF practices develop progressively deeper compliance intuition, producing requirements, designs, and code that naturally satisfy certification criteria without requiring post-hoc remediation, a learning effect that Hullmann and colleagues [13] document as one of the most durable benefits of sustained Agile-safety integration.

7. FUTURE EVOLUTION

HACF establishes a foundation for several near-term and longer-term extensions that will become increasingly relevant as avionics software complexity and regulatory expectations continue to evolve[23].

AI-assisted certification analytics represent the most immediate extension opportunity. Machine learning models trained on historical program compliance data can provide predictive CRI forecasts, identifying sprints at elevated finding risk before they close and enabling preemptive resource reallocation. Anomaly pattern recognition applied to traceability matrices can surface latent coverage gaps that rule-based consistency checks do not detect [24].

Automated compliance validation, beyond the current automated consistency checking described in Section 4.6, will extend to natural language processing of requirements artifacts to detect ambiguity, incompleteness, and non-testability before human review, reducing the certification engineer's review burden and improving the quality signal per review cycle [25].

Digital engineering integration will connect HACF compliance monitoring to model-based systems engineering environments, enabling traceability links to be maintained at the system model level and propagated automatically to software-level artifacts when system architecture changes occur, eliminating a class of traceability gap that currently requires manual reconciliation [26].

Predictive certification risk modeling will use program historical data and fleet-wide certification outcome databases to calibrate DAL-specific risk weightings in the CRI formula dynamically across program sprints, producing a risk-adaptive compliance management capability that improves with each program cycle.

CONCLUSION

This article has presented the Hybrid Agile-Certification Framework, an integrated execution model that resolves the structural incompatibility between iterative Agile software development and the evidence-

intensive compliance requirements of DO-178C avionics certification. HACF achieves this resolution not by relaxing certification requirements but by restructuring the program lifecycle so that compliance activities are embedded within every sprint, producing a continuously maturing compliance baseline that eliminates the late-stage defect accumulation and remediation cycles that undermine schedule, cost, and quality performance in traditionally structured avionics programs. The framework's five key components, iterative certification lifecycle, continuous shift-left certification, real-time compliance monitoring, risk-based verification, and cross-functional squad integration, work in concert to transform certification from a final program phase into a continuous program attribute. The quantitative evidence assembled in this article demonstrates that HACF-aligned execution produces outcomes that are not incrementally better than the traditional model but categorically superior across the most consequential program performance dimensions. SOI finding rates reduced by 70 to 83 percent, defect detection shifted 8 to 14 weeks earlier, traceability completeness improved by 20 to 30 percentage points, and pre-audit remediation effort reduced by 70 to 80 percent are outcomes that collectively address the root cause of the avionics certification performance problem rather than its symptoms. These improvements are structurally reproducible across programs that commit to the organizational integration, toolchain integration, and planning disciplines that HACF specifies, and they compound across program generations as cross-functional teams develop deeper compliance intuition through sustained practice.

Future research should extend the HACF evidence base in two directions. First, longitudinal studies of programs that have sustained HACF practices across multiple consecutive release cycles would illuminate the compounding efficiency gains that cross-functional learning produces and would provide more granular data on the DAL-specific verification burden characteristics of the framework. Second, the application of HACF principles to programs simultaneously pursuing certification under DO-178C and hardware assurance under DO-254 presents an integration challenge that the current framework does not fully address and that warrants dedicated investigation. The broader implication of this work is that safety-critical certification and iterative development methodology are not fundamentally incompatible; they require only a deliberate architectural integration that treats compliance as a first-class engineering attribute rather than a phase-end administrative burden.

REFERENCES

[1] RTCA, DO-178C: Software Considerations in Airborne Systems and Equipment Certification, 2011. Available: https://www.rtca.org/do-178/

[2] SAE International, ARP4754A: Guidelines for Development of Civil Aircraft and Systems, 2010. Available: https://www.sae.org/standards/arp4754b-guidelines-development-civil-aircraft-systems

[3] SAE International, ARP4761: Safety Assessment Process for Civil Airborne Systems, 1996. Available: https://www.sae.org/standards/arp4761-guidelines-methods-conducting-safety-assessment-process-civil-airborne-systems-equipment

[4] RTCA, DO-330: Software Tool Qualification Considerations, 2011. Available: https://my.rtca.org/productdetails?id=a1B36000001IcfkEAC

[5] Beck K. et al., Manifesto for Agile Software Development, 2001. Available: https://agilemanifesto.org

[6] Baron C., Louis V., Towards a continuous certification of safety-critical avionics software, Computers in Industry, vol. 125, p. 103382 (2021), DOI:10.1016/j.compind.2020.103382

[7] Heeager L.T., Nielsen P.A., A conceptual model of agile software development in a safety-critical context, Information and Software Technology, vol. 103, pp. 22-39 (2018), DOI:10.1016/j.infsof.2018.06.004

[8] Barbareschi M., Barone S., Carbone R., Casola V., Scrum for safety: an agile methodology for safety-critical software systems, Software Quality Journal, vol. 30, no. 4, pp. 1067-1088 (2022), DOI:10.1007/s11219-022-09593-2

[9] Islam G., Storer T., A case study of agile software development for safety-critical systems projects, Reliability Engineering and System Safety, vol. 200, p. 106954 (2020), DOI:10.1016/j.ress.2020.106954

[10] Zakrzewski P., Narkiewicz J., Brenchly D., Safety Critical Software Development Methodologies in Avionics, Transactions on Aerospace Research (2020), DOI:10.2478/tar-2020-0011

[11] Rhouas I., Nafil K., Avionic Software and Agile Development: A Systematic Mapping Study, World Conference on Information Systems and Technologies, Springer (2022), DOI:10.1007/978-3-031-04819-7_6

[12] Zampetti F. et al., Continuous integration and delivery practices for cyber-physical systems, ACM Transactions on Software Engineering and Methodology, vol. 32, no. 3, pp. 1-44 (2023), DOI:10.1145/3571854

[13] Hullmann J.A., Kimathi K., Weritz P., Large-scale agile project management in safety-critical industries, Information Systems Management, vol. 42, no. 2, pp. 138-160 (2025), DOI:10.1080/10580530.2024.2349886
[14] Ribeiro J.E.F., Silva J.G., Aguiar A., Scrum4DO178C: an Agile Process to Enhance Aerospace Software Development for DO-178C Compliance, IEEE Access (2025), DOI:10.1109/ACCESS.2025.3559803
[15] Baron C., Louis V., Esteve D., Framework and tooling proposals for Agile certification of safety-critical embedded software in avionic systems, Computers in Industry, vol. 148, p. 103887 (2023), DOI:10.1016/j.compind.2023.103887
[16] Maqsood H.M. et al., A Systematic Literature Review on Application of Agile SDPMs for Safety-Critical Systems, IET Software (2025), DOI:10.1049/sfw2/5227350
[17] Sartaj H. et al., Search-Based MC/DC Test Data Generation With OCL Constraints, Software Testing Verification and Reliability, vol. 35, no. 1 (2025), DOI:10.1002/stvr.1906
[18] Rempel P., Mader P., Preventing defects: The impact of requirements traceability completeness on software quality, IEEE Transactions on Software Engineering, vol. 43, no. 8, pp. 777-797 (2016), DOI:10.1109/TSE.2016.2622264
[19] Yahya N., Maidin S.S., The Waterfall Model with Agile Scrum as the Hybrid Agile Model, IEEE CITSM (2022), DOI:10.1109/CITSM56380.2022.9936036
[20] Uludag O. et al., Revealing the state of the art of large-scale agile development research, Journal of Systems and Software, vol. 194, p. 111473 (2022), DOI:10.1016/j.jss.2022.111473
[21] Mishra A., Alzoubi Y.I., Structured software development versus agile software development: a comparative analysis, International Journal of System Assurance Engineering and Management, vol. 14, no. 4, pp. 1504-1522 (2023), DOI:10.1007/s13198-023-01958-5
[22] Hanssen G.K., Stalahne T., Myklebust T., SafeScrum: Agile Development of Safety-Critical Software, Springer International Publishing (2018). Available: https://link.springer.com/book/10.1007/978-3-319-99334-8
[23] Manam K.B., An Exploration of Gender Biases and Discrimination Manifesting on AI/ML Development Team Dynamics, Doctoral Dissertation, University of the Cumberlands (2025). Available: University of the Cumberlands Repository.
[24] Nellutla N., Secure DevSecOps Workflows for Medical IoT Device Integration in Smart Hospitals, International Journal of AI, BigData, Computational and Management Studies, vol. 3, no. 1, pp. 114–122, (2022). Available: https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I1P113
[25] Hassan S.Z., Alang K., Nellutla N., Balasubramanian S.A., Morusu V., GovernAI: Policy-Driven Model Governance for Dynamic and Multi-Tenant AI Systems, 2025 International Conference on Information, Implementation, and Innovation in Technology (I2ITCON), pp. 1–8, (2025). Available: https://doi.org/10.1109/I2ITCON65200.2025.11210544
[26] Gollapudi R., Operational Drift and Risk-Bounded Decision-Making in Production Database Systems, Journal of International Crisis and Risk Communication Research, pp. 132–147, (2023). Available: https://doi.org/10.63278/jicrcr.vi.3762