Bridging Certification and Agile Development in Safety Critical Airborne Software: An Incremental DO-178C Assurance Framework

Prasannakumar Arumugam

Collins Aerospace Rockford, IL, USA

DOI: https://doi.org/10.51244/IJRSI.2026.1303000229

Received: 30 March 2026; Accepted: 04 April 2026; Published: 20 April 2026

ABSTRACT

Iterative development methods are increasingly adopted in safety-critical airborne software programs to address growing system complexity and evolving integration needs. At the same time, certification requires strict adherence to DO-178C assurance objectives [1][2].

Although DO-178C does not mandate a specific lifecycle structure, certification activities are frequently executed using milestone-oriented processes [1]. This introduces structural misalignment with incremental development, often leading to delayed compliance evidence, increased rework, instability in verification artifacts, and uncertainty during certification reviews [7][9].

This paper proposes an Incremental Assurance Framework that integrates certification objectives directly into iterative workflows. The framework is based on three key elements: progressive fulfillment of assurance objectives, disciplined handling of change impacts, and explicit maintenance of verification independence.

A four-level Agile–Certification Maturity Model is also presented to describe how organizations evolve toward sustained certification readiness. A representative scenario is used to illustrate improvements in trace consistency, reduced review disruption, and enhanced predictability of certification outcomes.

The results indicate that certification activities can be performed continuously alongside development while maintaining the rigor required for airborne software approval.

Keywords: DO-178C, airborne software, Agile methods, safety-critical systems, certification strategy, incremental assurance, verification independence, traceability.

INTRODUCTION

Modern airborne platforms rely heavily on embedded software to implement functional behavior, system coordination, and safety mechanisms. Demonstrating correct operation within the intended environment is essential for certification [1][7].

DO-178C defines a set of assurance outcomes that must be satisfied through verifiable evidence [1][2]. While the standard does not prescribe a particular development sequence, certification practices have historically been implemented using structured, phase-based approaches aligned with major review milestones [1][3].

In parallel, iterative development methods have gained attention due to their ability to support evolving requirements, faster integration, and improved collaboration [9]. These approaches emphasize incremental delivery, continuous feedback, and adaptive planning.

The difficulty arises when iterative development is combined with certification practices that assume consolidated evidence at predefined milestones. This mismatch often results in gaps between development progress and certification readiness, leading to late-stage corrections and increased program risk [7][10].

In addition, coordination between system engineering and software development activities can become fragmented when both operate under different cadences and governance models. This misalignment affects requirement decomposition, trace relationships, and verification completeness.

This paper addresses the following question:

How can iterative development be organized such that certification objectives are satisfied continuously without reverting to rigid sequential processes?

To address this challenge, a structured framework is proposed that aligns development activities with certification expectations at the iteration level. Specifically, this work introduces an incremental assurance approach for embedding certification objectives within iterative workflows, defines structural mechanisms to maintain compliance discipline and verification independence, and presents a maturity model to characterize progression toward continuous certification readiness.

Unlike prior approaches that focus on lifecycle adaptation or tool-based augmentation, the proposed framework introduces a process-centric integration of DO-178C assurance objectives directly within iterative execution. This enables continuous certification readiness independent of specific lifecycle models or toolchains, addressing a key gap in existing Agile–certification integration strategies.

The proposed Incremental Assurance Framework and the Agile–Certification Maturity Model are illustrated in Fig. 1 and Fig. 2, respectively.

The proposed framework is derived from synthesized industry observations and generalized practices across safety-critical software programs.

This paper makes the following contributions:

1. Proposes an Incremental Assurance Framework integrating DO-178C objectives into Agile workflows
2. Defines structured mechanisms for maintaining verification independence in iterative environments
3. Introduces an Agile–Certification Maturity Model for organizational assessment
4. Identifies common integration risks and mitigation strategies based on industry observations

Do-178c as an Objective-Based Assurance Framework

DO-178C defines a set of assurance objectives intended to ensure that airborne software performs its intended functions within its operational context [1][2]. Rather than prescribing a specific development sequence, the standard defines a set of assurance outcomes that must be satisfied and supported with verifiable evidence [1], [4].

Key assurance outcomes include:

- Controlled and clearly defined software requirements [1], [2]
- Demonstrable traceability relationships across requirements, design, code, and verification artifacts [1], [4]
- Verification activities scaled according to software criticality [1], [2], [4]
- Evidence that testing sufficiently exercises implemented logic [1], [2]
- Formal configuration control of lifecycle data [1], [2]
- Independent oversight of compliance processes [1], [4]

- Structured coordination with certification authorities [1], [7]

The rigor applied to these outcomes increases with software level, reflecting the potential safety impact of failure conditions [1]. Importantly, DO-178C evaluates whether required objectives are satisfied—not whether development follows a particular phase order [1], [2]. Sequential lifecycle models have historically been adopted to organize evidence, but they are not regulatory mandates [1]. Many perceived incompatibilities between Agile and certification arise from process habits rather than regulatory constraints [9].

Recognizing DO-178C as an **outcome-driven assurance model** is fundamental to integrating it with iterative development practices [2], [8]. The scholarly article "Ensuring Compliance with DO-178C: Advanced Techniques in Avionics Software Verification," by Jawahar Thangavelu, examines the rigorous standards for airborne software. It focuses on the verification methods required to meet functional safety and reliability criteria essential for certifying safety-critical systems [12].

Observed Integration Risks in Agile Certification Programs

Experience across multiple airborne software efforts reveals recurring structural risks when Agile execution is introduced without explicit certification integration [10][11].

Certification Debt Accumulation

Functional increments are delivered rapidly, while compliance artifacts trail behind [3]. Trace updates, verification records, and coverage analyses are deferred to later integration cycles.

Consequences include:

- Divergence between implemented behavior and documented traceability [1][4]
- Increased corrective action prior to certification reviews [3]
- Elevated review volatility [7]

Independence Dilution

Cross-functional sprint teams can unintentionally weaken verification independence. Peer reviews within the same reporting structure may not satisfy higher assurance expectations unless independence is demonstrable structurally, not merely culturally [1][4][9].

Structural Coverage Instability

As logic evolves incrementally, structural coverage results may vary across successive builds [1][4]. Coverage achieved in early builds may regress following refactoring or integration merges. Without synchronized tracking against stable baselines, apparent completeness can be misleading.

Change Impact Underestimation

Iterative backlog refinement may treat changes as localized adjustments [9]. In tightly coupled embedded architectures, modifications can propagate across modules and interfaces. Insufficient impact analysis leads to incomplete regression scope [10][11].

Toolchain Governance Volatility

Continuous integration environments rely heavily on automation. Frequent tool updates, if not governed, can undermine previously established qualification assumptions and introduce compliance instability [4][7].

System–Software Integration Misalignment

A recurring challenge in Agile adoption within safety-critical airborne programs is the misalignment between system engineering and software development activities. System-level requirements are often defined, decomposed, and managed within separate organizational and process structures, frequently operating with different lifecycle cadences and traceability mechanisms.

In the absence of structured integration, Agile iterations at the software level may proceed without synchronized alignment to evolving system requirements. This disconnect can result in:

- Incomplete or inconsistent requirement decomposition across system and software levels
- Gaps in end-to-end traceability between system requirements, software design, and verification artifacts
- Delayed identification of integration and interface issues
- Increased certification risk due to fragmented and non-cohesive compliance evidence

As a result, organizations may revert to sequential development approaches to regain control over traceability and verification alignment, thereby limiting the effective adoption of Agile practices in safety-critical environments.

Comparison with Existing Approaches

Existing approaches to integrating Agile development with safety-critical certification generally fall into three categories. The proposed framework differs by focusing on process-level integration of assurance objectives, ensuring that compliance is continuously achieved independent of lifecycle model or tooling strategy.

Table I: Comparison of Agile–Certification Integration Approaches

Approach	**Key Characteristics**	**Limitations**	**Incremental Assurance Framework Advantage**
Traditional DO-178C Lifecycle	Sequential, milestone-driven compliance	Late validation, high rework risk	Enables continuous objective closure
Generic Agile Adoption	Iterative development without certification mapping	Certification gaps, compliance debt	Embeds assurance into iteration workflow
Model-Based Approaches (DO-331)	Tool-supported development and verification	Tool dependency, integration complexity	Process-driven, tool-agnostic integration

Incremental Assurance Framework

To address these risks, a structured framework is proposed based on three pillars [9][10]. Fig. 1 illustrates how the Incremental Assurance Framework integrates certification objectives within Agile iterations to achieve continuous compliance.

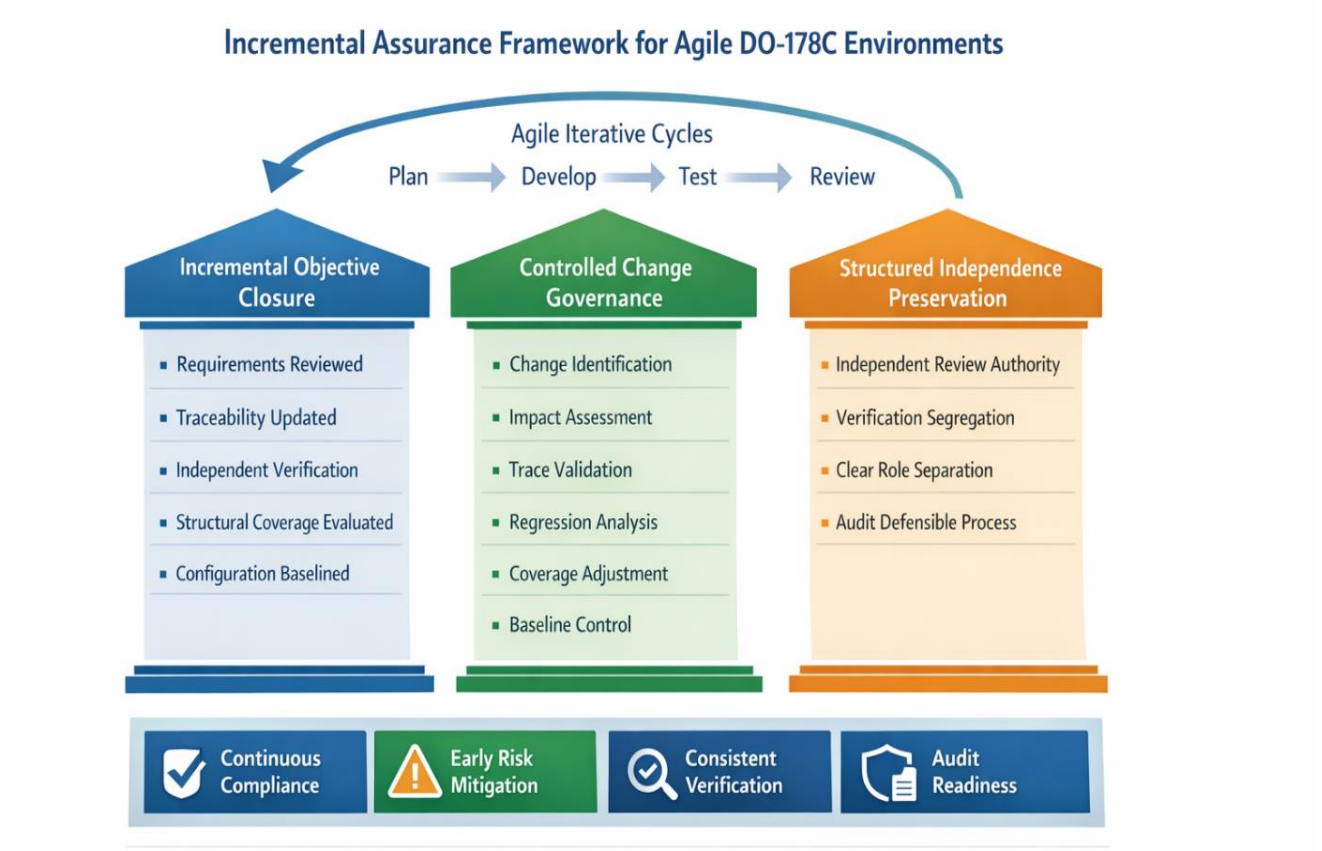

Fig. 1. Incremental Assurance Framework integration of certification objectives with Agile iteration, illustrating continuous compliance achievement.

Fig. 1 emphasizes the continuous alignment between development increments and certification objectives, demonstrating how assurance activities are performed within each iteration rather than deferred to milestone phases.

Pillar I – Incremental Objective Closure

Each iteration must close applicable assurance objectives associated with the functionality introduced during that increment [1][2][4].

A certification-aware Definition of Done includes:

- Requirements formally reviewed and controlled [1][2]
- Trace relationships updated and validated [1][4]
- Code compliant with defined standards [1][2]
- Independent verification completed [1][4]
- Structural coverage impact evaluated [1][4]
- Configuration items baselined [1][2]

Compliance becomes a continuous activity rather than a deferred consolidation phase [9][10].

Pillar II – Controlled Change Governance

Requirement and design modifications must flow through a structured impact workflow integrated with Agile backlog processes [9][10]:

1. Formal change identification
2. Cross-artifact impact assessment
3. Trace validation update
4. Regression scope determination
5. Coverage delta evaluation
6. Configuration baseline adjustment

Automation can support scalability, but safety-relevant judgment requires independent review [1][4][7].

Pillar III – Structured Independence Preservation

Verification independence must be demonstrable through organizational structure and authority lines [1][4][9].

Mechanisms may include:

- Clear reporting separation between development and verification [1][4]
- Independent approval authorities [1][7]
- Defined compliance checkpoints outside sprint ceremonies [9]
- Documented role segregation [10]

Independence must be structurally defensible during audit, not inferred from team collaboration norms [1][4].

Agile–Certification Maturity Model

Organizational integration of Agile and certification typically evolves progressively [9][10]. The model defines four progressive stages of organizational maturity. Fig. 2 illustrates the proposed Agile–Certification Maturity Model enabling continuous compliance through progressive integration of assurance objectives within Agile workflows for safety-critical airborne software.

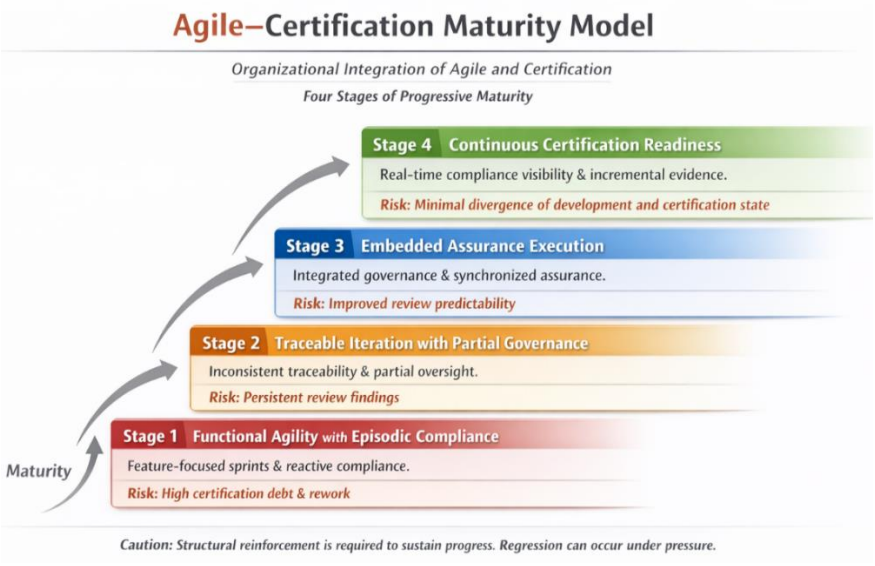

Fig. 2. Proposed Agile–Certification Maturity Model enabling continuous compliance through progressive integration of assurance objectives within Agile workflows for safety-critical airborne software.

The model highlights the transition from reactive compliance practices to continuous certification readiness, with increasing levels of assurance integration and organizational discipline.

Stage 1 – Functional Agility with Episodic Compliance

Iterations focus on feature delivery. Compliance evidence is assembled near major review milestones [9].

Risk: High certification debt and late rework.

Stage 2 – Traceable Iteration with Partial Governance

Trace updates occur within iterations, but independence enforcement and impact discipline remain inconsistent [10].

Risk: Reduced but persistent review findings.

Stage 3 – Embedded Assurance Execution

Assurance objectives form part of iteration exit criteria. Change governance and coverage tracking are synchronized with integration baselines [1][4][9].

Risk: Significantly improved review predictability.

Stage 4 – Continuous Certification Readiness

Compliance posture is visible at all times. Evidence evolves incrementally alongside development. Major review events exhibit low volatility [10].

Risk: Minimal divergence between development state and certification posture.

Sustained progression requires structural reinforcement. Under schedule pressure, regression is possible if objective closure discipline weakens [9][10].

Illustrative Composite Scenario

In a representative embedded airborne subsystem, initial Agile adoption prioritized rapid feature integration but lacked structured compliance embedding. Additionally, misalignment between system-level requirement management and software iteration cycles contributed to trace inconsistencies and integration gaps. These challenges, combined with coverage instability, emerged during early review assessments [10][11].

Adoption of incremental objective closure introduced:

- Formalized sprint exit criteria aligned with assurance outcomes [1][2]
- Independent verification sign-off external to sprint planning [1][4]
- Continuous coverage monitoring tied to integration baselines [1][4]
- Segregated governance of automation tools [4][7]

Subsequent review cycles showed improved evidence coherence, reduced corrective action volume, and improved schedule predictability [9][10].

Following implementation, subsequent review cycles demonstrated measurable improvements:

- Reduction in certification review findings and associated corrective actions (observed improvement trends across iterative cycles)

- Improved traceability consistency across lifecycle artifacts
- Stabilization of structural coverage metrics across integration baselines
- Reduced volatility during SOI assessments

These outcomes indicate that embedding assurance objectives within iterative workflows can improve certification predictability and reduce late-stage rework.

These observations are based on aggregated behavior across multiple iterative development cycles within representative safety-critical embedded software environments, reflecting consistent directional improvements rather than isolated outcomes.

Quantitative Indicators of Framework Effectiveness

While absolute values vary across programs, representative ranges observed across iterative cycles indicate meaningful improvement. In representative observations across iterative cycles, certification review findings exhibited reductions on the order of 20–40% relative to initial baseline reviews. These values are indicative and may vary depending on system complexity, assurance level, and organizational maturity. For example, traceability discrepancies decreased significantly following enforcement of incremental objective closure.

Table II summarizes indicative trends associated with the adoption of incremental assurance practices.

Table II: Representative Trends in Certification Metrics Following Framework Adoption

Metric	**Pre-Adoption Characteristics**	**Post-Adoption Characteristics**	**Observed Trend**
Certification Review Findings	High volume of findings during SOI reviews	Reduction in certification review findings (observed decreasing trend across successive SOI cycles compared to initial baseline reviews)	Decreasing trend
Traceability Consistency	Frequent gaps and late updates	Continuous alignment across artifacts	Significant improvement
Structural Coverage Stability	Fluctuations across builds	Stable coverage aligned with baselines	Improved stability
Late-Stage Rework Effort	Concentrated near certification milestones	Distributed across iterations	Reduced rework concentration
SOI Review Volatility	High variability in review outcomes	Consistent and predictable outcomes	Reduced volatility

To characterize certification stability trends, a normalized metric can be defined as:

Certification Stability Index (CSI) = (F_baseline − F_iteration_n) / F_baseline

where F represents the number of certification review findings. This metric provides a relative measure of improvement across iterative cycles and can support comparative assessment across programs.

These observations indicate that embedding assurance objectives within iterative workflows contributes to progressive stabilization of certification artifacts, improved trace completeness, and reduced uncertainty during certification reviews.

While absolute values vary across programs, the consistency of directional improvement across multiple metrics indicates measurable impact of incremental assurance practices.

Organizational Benefits

When properly implemented, the Incremental Assurance Framework provides:

- Reduced late-stage compliance rework [9][10]
- Lower volatility at SOI reviews [1][7]
- Improved audit defensibility [1][4]
- Stable structural coverage posture [1][4]
- Stronger independence demonstration [1][4]
- Greater program predictability [9][10]

Agile execution becomes an enabler of structured assurance rather than a source of compliance instability [8][9].

DISCUSSION AND PRACTICAL IMPLICATIONS

The proposed framework is derived from generalized industry practices and observed integration challenges across safety-critical airborne software programs.

Its effectiveness is supported by observed improvements across iterative development cycles, including reduction in certification review findings, improved traceability consistency, and stabilization of structural coverage metrics.

While the illustrative scenario provides qualitative validation, future work will focus on empirical evaluation through controlled case studies and quantitative assessment of certification effort, defect density, and review efficiency.

The framework is intended to be adaptable across different organizational contexts, independent of specific tools or lifecycle models.

Toolchain and Certification Authority Considerations

Effective implementation of incremental assurance practices requires disciplined governance of development tool chains and structured engagement with certification authorities.

Continuous integration environments introduce dependencies on automated tools for build, verification, and analysis activities. To maintain compliance stability, toolchains must be governed through controlled baselines aligned with applicable tool qualification guidance. Changes to tool versions, configurations, or analysis mechanisms should be subject to formal impact assessment to ensure that previously established assurance evidence remains valid.

Key toolchain governance considerations include:

- Controlled configuration baselines for qualified and non-qualified tools
- Impact assessment for tool updates affecting verification results

- Traceability between tool outputs and certification artifacts
- Periodic validation of toolchain behavior within integration environments

In parallel, certification authority engagement must evolve to align with iterative development practices. Rather than deferring interactions to major milestones, incremental assurance supports progressive visibility into compliance status.

This approach enables early alignment of compliance interpretation, reducing the likelihood of late-stage certification findings during SOI reviews.

Recommended practices include:

- Incremental sharing of certification evidence aligned with iteration outputs
- Early engagement with certification authorities prior to formal SOI reviews
- Periodic readiness assessments reflecting current compliance posture
- Alignment of iteration-level assurance activities with SOI expectations

Continuous and structured engagement enables early validation of compliance artifacts, reduces uncertainty during formal reviews, and supports a more predictable certification process.

These practices align with tool qualification expectations defined in DO-330 and support sustained compliance across evolving development environments.

Adoption in Resource Constrained Environments

The proposed framework assumes a level of organizational discipline and process maturity that may not be immediately achievable in all contexts. However, incremental adoption strategies can enable practical implementation in resource-constrained or lower-maturity environments.

Organizations may adopt the framework progressively by prioritizing foundational assurance practices before expanding to full integration. Initial focus areas may include establishing consistent traceability, implementing structured change impact analysis, and enforcing basic configuration control mechanisms.

A phased adoption approach may include:

- Establishing end-to-end traceability across requirements, design, and verification artifacts
- Introducing formal change impact assessment within backlog refinement processes
- Gradually strengthening verification independence through role separation and review structures
- Incrementally integrating coverage tracking and automation support

By scaling implementation based on program size and criticality, organizations can realize incremental benefits without requiring immediate large-scale transformation. This approach reduces adoption barriers while enabling gradual progression toward continuous certification readiness.

Limitations

The proposed framework is derived from synthesized industry observations rather than controlled experimental studies. The quantitative improvements presented are indicative and may vary depending on system complexity, assurance level, and organizational maturity. Additionally, the framework assumes a level of process discipline that may not be immediately achievable in lower-maturity or resource-constrained environments. Future work

will focus on empirical validation through controlled case studies and statistical evaluation of certification outcomes.

CONCLUSION

The perceived incompatibility between Agile development and DO-178C certification arises primarily from historical lifecycle interpretations rather than regulatory constraints [1][2].

DO-178C defines assurance objectives that can be satisfied within iterative environments when embedded structurally within development workflows [1][4][9]. By implementing incremental objective closure, disciplined change governance, and preserved independence, organizations can achieve adaptive development while maintaining regulatory integrity.

Beyond current avionics programs, the proposed framework has broader applicability to emerging domains such as autonomous systems, advanced air mobility platforms, and next-generation integrated avionics architectures, where iterative development and continuous integration are increasingly prevalent.

By enabling continuous certification readiness, the Incremental Assurance Framework provides a scalable foundation for aligning safety-critical software assurance with modern development paradigms, supporting both regulatory compliance and innovation in aerospace systems.

This work contributes to ongoing efforts within the aerospace community to modernize certification practices in alignment with evolving software development paradigms.

This work demonstrates that certification and Agile development are not inherently conflicting paradigms, but can be systematically unified through structured assurance integration, enabling both regulatory rigor and development agility.

Disclaimer

The concepts and scenarios presented in this paper are generalized and anonymized to preserve confidentiality. This work is intended to contribute to broader industry understanding of certification–Agile integration in safety-critical avionics systems.

REFERENCES

1. RTCA, DO-178C: Software Considerations in Airborne Systems and Equipment Certification, 2011.
2. EUROCAE, ED-12C: Software Considerations in Airborne Systems, 2011.
3. RTCA, DO-330: Software Tool Qualification Considerations, 2011.
4. RTCA, DO-331: Model-Based Development and Verification Supplement to DO-178C, 2011.
5. RTCA, DO-332: Object-Oriented Technology Supplement to DO-178C, 2011.
6. RTCA, DO-333: Formal Methods Supplement to DO-178C, 2011.
7. FAA, AC 20-115D: Airborne Software Development Assurance Using DO-178C, 2017.
8. Leveson, N., Engineering a Safer World: Systems Thinking Applied to Safety, MIT Press, 2011.
9. Highsmith, J., Agile Project Management: Creating Innovative Products, Addison-Wesley, 2010.
10. IEEE Aerospace Conference, Selected Proceedings on Agile Methods in Safety-Critical Systems, 2015–2022.
11. Parnas, D., & Madey, J., “Software Aging: The Evidence and Implications for Safety-Critical Systems,” Journal of Systems and Software, 2009.
12. Thangavelu, Jawahar, Ensuring Compliance with DO-178C: Advanced Techniques in Avionics Software Verification (January 06, 2022). ESP Journal of Engineering & Technology Advancements ,Volume 2 Issue 1 February 2022 / Page No: 135-146, 10.56472/25832646/JETA-V2I1P116, Available at SSRN: https://ssrn.com/abstract=5124611