

Original Article

# Ensuring Compliance with DO-178C: Advanced Techniques in Avionics Software Verification

**Jawahar Thangavelu**

Software Engineer, USA.

Received Date: 06 January 2022

Revised Date: 03 February 2022

Accepted Date: 28 February 2022

**Abstract:** The DO-178C standard includes specific rules for software implemented in airborne systems to meet aviation's functional safety and reliability criteria. This paper focuses on some modern forms of avionics software verification following DO-178C. Stress is made on how new-age verification techniques, such as model-based design, static analysis, automatic test generation, and formal verification, work in the real world. The document also assesses the performance of such techniques based on how problems related to safety-critical aspects are solved and how the traceability and certification times are optimized. The proposed framework incorporates the current best practices in verification together with the compliance processes to ensure that safety is not compromised while efficiency is improved. These approaches are illustrated by case studies and experimental results, revealing practical approaches to attaining certifiable software with a small number of flaws. **Keywords:** DO-178C, Avionics Software, Verification, Certification, Automated Testing, Formal Methods.

**Keywords:** DO-178C, Avionics Software, Verification, Certification, Automated Testing, Formal Methods.

## I. INTRODUCTION

Avionic computer systems are safety-critical software that, if failed, can lead to severe consequences that are often fatal. DO-178C is the main standard for software in airborne systems and consists of recommendations on software planning, development, verification, and configuration management. [1-3] Due to its complex demands, its compliance requires using unique verification techniques.

### A. Role of Avionics Software Verification

Avionics software is necessary because it makes aircraft systems safe, reliable and functional. Since the introduction of advanced aviation technologies, there has been an ever-increasing need for reliable means for avionics software verification. Avionics software verification ensures it runs appropriately and securely and meets comprehensive benchmarks such as DO-178C. This section elaborates on the role of avionics software verification through the following subheadings:



**Figure 1: Role of Avionics Software Verification**



a) *Ensuring Safety and Reliability:*

Aerospace software verification means that software for controlling the plane, navigation, and communication is operational and does not insert risks. Processes like static analysis and formal methods help identify weaknesses and guarantee that the program functions correctly, no matter the circumstances. This helps minimize failures that would lead to disastrous incidents in any production facility.

b) *Compliance with Regulatory Standards:*

The DO-178C was developed to qualify avionics software for use in the commercial aviation industry. The standard prescribes detailed procedures for software development, verification and validation to meet the safety and functionality stipulated in the standard. Guaranteeing mechanisms, such as traceability and sound testing, are critical in ensuring that one is able to prove that software deployment indeed complies with these regulations.

c) *Detecting and Preventing Defects:*

The early identification of defects is important in order to maintain the reliability of avionics software where faults can cause significant problems. Static analysis and automated testing methods used for code examination make it possible to detect many issues and prevent their escalation, significantly reducing the cost of defect rectification. This makes it possible to avoid product failures during its operation and guarantees that this software meets safety requirements.

d) *Enhancing Software Quality:*

Reliability, safety and efficiency of avionics depend greatly on the quality of software that is used in the flight. Such methods include Code reviews, proofreading and/or automated testing, which helps in detecting some wrongs, such as bad coding standards and construction. There is additional improvement to quality through the use of formal methods that show the right mathematical aspects that ensure the system's core components are right.

e) *Reducing Certification Time and Cost:*

There are clear correlations between main verification processes and the time and cost of certification for avionics software. Thus, through test automation and the application of methodologies such as model-based design and static analysis, the verification task is solved more effectively. This cuts down on the work done manually, helps identify defects faster, and shortens the time taken for certification besides meeting the required standards.

f) *Managing Complexity in Safety-Critical Systems:*

Progressing technological densities in avionic implementations require reliable methodologies to analyze intricate designs and enormous volumes of code. This is where Model-based design (MBD) comes in handy as it operates on a different plane to minimize manual coding mishaps. In addition, rigorous approaches also guarantee that the key system components are proven mathematically for their correctness and to manage the complexity that is inherent in the current systems.

g) *Supporting Continuous Integration and Development:*

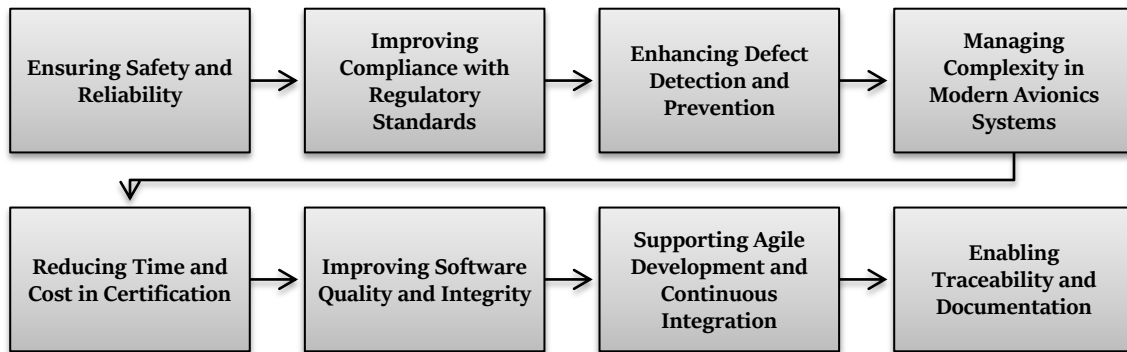
Since Agile and DevOps have been embraced, avionic software verification has to integrate per continuous integration and development (CI/CD). This is a rapid way of testing software changes since there is a constant check to ensure that whatever is developed is perfect without having to wait for a long time to beta test the product. This approach ensures avionics software's safety, reliability, and conformance to standards up to the time of deployment.

h) *Traceability and Documentation:*

Verification is an important part of DO-178C; it says that all the requirements for the software must be traceable and documented. Other verification tools assist in making traceability more or less automatic by matching requirements to test cases and design elements. Documentation simplifies the issue of certifications as it offers a record that shows that the processes followed agree with best practices.

## **B. Importance of Advanced Techniques in Avionics Software Verification**

System integration and verification of avionics software is important in order to gain assurance that the software being used is safe, reliable, and meets all standards necessary as a system of record. [4,5] When avionics systems grow complicated and are used to facilitate safety-critical applications, the conventional verification approaches are inadequate. The latest tools offer better and more efficient ways of addressing stringent software quality standards such as DO-178C. This section discusses the need for high-level techniques like model-based design, static code analysis, formalism and testing in avionic software system development.



**Figure 2: Importance of Advanced Techniques in Avionics Software Verification**

*a) Ensuring Safety and Reliability:*

Avionics software verification pinpoints flaws in safety-relevant systems, such as flight control and navigation, to prevent the deployment of hazardous errors. Sophisticated large-scale methods like static code analysis assist in identifying risks that root back to failures such as Buffer overflow, uninitialized variables and many more. Techniques based on mathematics have been employed to guarantee the specified safety and reliability of the system and decrease the possibility of a system failure.

*b) Improving Compliance with Regulatory Standards:*

DO-178C, in particular, provides guidance on avionics software certification, and it provides great emphasis on verification. Sophisticated methods include model-based design, which reduces compliance problems by generating code directly from models. Automated testing adds to compliance in even more ways by generating massive test cases, reaching likely functional and edge cases, and producing documentation that will meet DO-178C guidelines.

*c) Enhancing Defect Detection and Prevention:*

Enhancement verification methods enhance defects discovered through the early definition of potential errors in the development phase to minimize the occurrence of failure later. Polyspace and Coverity are examples of Static code analysis tools that scan the code and pick out defects like memory leakages and boundary violations without running the code. The automated testing framework facilitates constant assessment of software, hence enabling developers to prevent the inclusion of defects during the development process.

*d) Managing Complexity in Modern Avionics Systems:*

When avionics systems become more complex, the verification technique has to deal with the larger volumes of code as well as the complex cross-modulation system. In model-based design (MBD), components are described at higher levels of abstraction where these behaviors are unambiguous and where the system interactions are simplified to their nominal forms. This, in turn, guarantees that all system elements are integrated and operate in the right manner, irrespective of the levels of complexity of the component parts.

*e) Reducing Time and Cost in Certification:*

Certification is an elaborate and expensive affair, but the enhanced verification procedures make it easier. Automated testing means the reduction of effort used for test case generation and test case execution, while model based design means the speeding up of generating code and models from models. Implementing these techniques automates manual tasks in certification, which in turn decreases certification time and cost and increases faster, cheaper product release.

*f) Improving Software Quality and Integrity:*

In avionics, retaining the highest quality of developed software is critical to achieve safety and good performance. Static code analysis and formal methods target both functional problems and coding and documenting standards and practices. Specification techniques have the additional advantage of being able to provide formal proofs of correct behaviors of important components of the software that run to specifications as designed.

*g) Supporting Agile Development and Continuous Integration:*

The gradual approach to application development and the CI/CD process requires constant assurance to decrease risks or issues with software reliability, safety, or quality. The modern form of verification, like automated testing, static analysis, etc.,

helps to maintain Agile because they offer feedback on a real-time basis on the changes made to the code. Incorporating such tools into the CI/CD escalates defects early enough to ensure that the final developed software does not fail to meet compliance and safety measures.

*h) Enabling Traceability and Documentation:*

Documentation requirements follow the traceability of software requirements, design and proofs in DO-178C. Sophisticated activities such as model-based engineering and test automation guarantee that each testing scenario is associated with requisite specifications, thereby ensuring complete traceability. This is made easier through tool support for automated documentation, which cuts out a lot of manual work while creating a clear paper trail that is greatly required for certification and maintenance of the software.

## II. LITERATURE SURVEY

### A. Overview of DO-178C

DO-178C, or Software Consideration in Airborne System and Equipment Certification, is the most important standard for avionics software certification in civil aircraft. The software is grouped into five safety levels: Level A (software failure, which might lead to catastrophic consequences), B, C, D and E (least critical). The standard requires each process involved in software development to undertake re-verification and validation. DO-178C, like most standards for software development, is centered on a requirements-based approach, which means that every item of software must be shown to meet its requirements. [6-10] This is complemented by traceability, where every requirement is linked to its design and verification product, enabling transparency through the software development life cycle. Moreover, DO-178C also connotes the concept of providing evidence for compliance, which asserts that every verification activity performed was done and documented to the extent of the standard. This evidence provides the grounds for certification by such authorities as the Federal Aviation Administration (FAA) or the European Union Aviation Safety Agency (EASA).

### B. Traditional Verification Methods

Research on avionics software or traditional software development shows that conventional software verification procedures have been in use for a long time. Manual code checks or code reviews, requirement inspection, and unit testing broadly apply these methods. An outdated code review is to have one programmer read through the code for defects, which can be tedious and inaccurate. Verification assures that software meets the design requirements laid down at the commencement of the project, usually by inspection or walkthrough. The purpose of unit testing is usually to check one or several isolated components of the software. Although these techniques have been effective in the past in most industries that have adopted them, they are not very effective in large and complex systems. Namely, traditional approaches do not allow for detecting such weaknesses in the early stages of development, and the manual testing approach may neglect certain important peculiarities or cover the system's capabilities insufficiently or inadequately. Moreover, these methods may require more time and are relatively gradual, contributing to the duration necessary to gain certification and negatively impacting the creation of safety-critical systems.

### C. Modern Verification Techniques

The recent developments in modern verification techniques make a mammoth improvement of the avionics software development in terms of reliability and efficiency, especially in compliance with DO-178 C. MBD uses system models to automatically produce code and associated verification, thereby minimizing the amount of human intervention and assuring design and code conformance. Programs such as MATLAB's Simulink enable the developer to create the actual application at an abstract level to enhance the speed of development and verify it accurately to allow for early simulations and testing. Polyspace and Coverity are types of Static Code Analysis that check source code without running it to detect defects, including memory leaks, buffer overflows and coding violations; however, they often give false anomalous readings that require old-fashioned manual examination. Above the verification process, we have Formal Methods that use mathematical models and proof outcomes for the correctness of software with high assurance for the critical systems, but these remain highly complex to implement and require high computation power for their implementation can only be effectively used for the critical components of a system. Test Complete is an example of the Automated Testing approach that automatically augments the test process by creating Test Cases; all the key and peripheral Use Cases are tested like functional and extreme test cases. Coupled with the continuous integration pipelines, this technique makes it easy to perform regression tests as the system changes. However, it is important to note that the effectiveness of automated testing is realized with reference to the test generation mechanisms that only provide sufficiently diverse and realistic inputs to effectively detect possible subtle faults. Together, these updated verification methods

provide a broader, more effective, and more efficient method of avionics software development that, in turn, enhances the way the DO-178C standard is met faster and more effectively.

#### **D. Challenges in Compliance**

That said, there are still several issues that one is likely to encounter while certification against DO-178C and other safety-related standards. One of the bigger issues is the ability to track back each requirement and identify that it has been traced to the design, implementation, and verification phases. This entails great attention to documentation and management, thus calling for cumbersome and tiresome techniques. The two key challenges are avionics and, more specifically, the complexity of the software used on current commercial aircraft. To this end, these systems are growing and have many interdependent components and sub-components, making it hard to prove all functionality at one given time. In addition, it would be extremely difficult to comprehensively test high-assurance systems for thoroughness since the number of possible combinational failures is immense, and many may not be obvious. As with the reliance on automated tools, there is also the question of tool qualification confirming that the end product and the tools themselves are correctly validated and verified. These challenges clearly indicate a need to overhaul the approach to verification involving both traditional and advanced means.

#### **E. Comparative Analysis of Techniques**

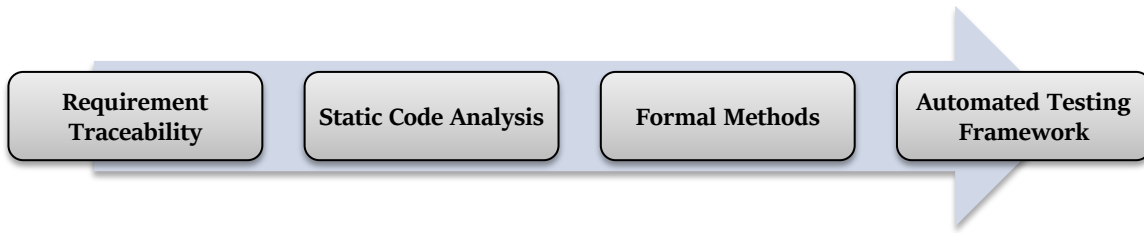
A comparative study of the latest software verification techniques shows that each method has unique advantages and deprecations. Static Code Analysis is excellent and very useful for the initial things like detecting obvious errors and code inspection all before actually running the application Polyspace, Coverity or the like for such things as uninitialized variables and so on - buffer overflows. Nevertheless, it must give out alerts that can give positive results that are mistaken, and these alerts will need to be checked by hand. Model-Based Design (MBD) incorporates fewer errors from the design procedure and saves time as it converts system models straight into code, expediting both design and verification. Nevertheless, several improvements should be mentioned. MBD requires extensive initial costs for the tools and training, and its utilization may be difficult even for experienced professionals because of the complexity of the concept. Formal Methods prove mathematically that the software developed is correct and, therefore, is well suited for applications in highly dependable real-time systems where a failure can be disastrous. However, they have many disadvantages mainly that ANN is more difficult to learn, and the use of significant amounts of computational resources makes it only feasible for critical system components. Automated testing is similar to Test Automation in that it enables the testing of requirements through automated testing tools that cycle through the test cases and generate tests that return the outcome of the tests within a short span. Nonetheless, its applicability is limited because test case generation capability may not be sophisticated enough to produce sufficient structural variation or realistic inputs for demonstrating that defects are shown inadequately. All the techniques have useful contributions in the course of verification. However, their drawbacks show the importance of using a system which incorporates the best from various techniques for efficiency and completeness of the verification process.

### **III. METHODOLOGY**

#### **A. Proposed Framework**

The proposed solution brings together a number of modern techniques for verification into a single process of checking the comprehensiveness of compliance with DO-178C. It underlines a collaborative strategy to enhance software verification tasks in safety-critical avionics systems by using requirement traceability, static code analysis, formal methods, and test automation. [11-15] The entire development process is initiated with requirement specification that links each of the requirements bidirectional to the development and verification products. Static code analysis is initially used to identify coding standards' violations and defects and reduce the expensive corrective actions at later phases. Software Verification is employed to formally prove the correctness of significant algorithms and establish high-reliability assurance for critical functions to high-assurance systems. With these, there are sophisticated test-benefiting automation systems that augment effectiveness as they automatically create a number of test-benefiting cases in relation to the requirements and provide comprehensive test coverage across each software part. Due to the modular and intuitive manner of incorporating these techniques into the proposed framework, the verification process becomes structured, the number and quality of defects identified are improved, and the time to certification is optimized to meet the requirements of the DO-178C standard for safety and quality.

## B. Key Components



**Figure 3: Key Components**

### a) Requirement Traceability:

Documentation requirement traceability is at the core of compliance with DO-178C, which mandates retrospective and prospective accounts for all accommodation requirements in development and verification activities. It means setting up bi-directional traceability links between requirements and other related design, implementation, and testing. This practice also helps teams to document how and to what extent each requirement is met, where problems and/or inconsistencies exist, and to prove satisfaction of certification requirements to certification authorities. In addition, traceability improves maintainability because it provides the means of correctly analyzing the ripple effect of changes on dependent elements of requirements documentation.

### b) Static Code Analysis:

Static is an early-time method of analyzing the source code to detect faults, weaknesses, and nonstandard code without the program's working. This automated stuff shows problems from the development stage, such as null pointer dereferences, resource leakage, and moving off of safety-critical coding standards such as MISRA or CERT. As these problems are addressed before they expand into the subsequent states, static code analysis lessens the revisions, decreases the expenses on development, and guarantees that the software features all the high degrees of quality to which DO-178C entitled it.

### c) Formal Methods:

Formal methods attempt to prove the adequacy and dependability of the critical algorithms in avionics software. These techniques employ formal proofs to affirm the accuracy of the functional specification of the software under all possible usage and conditions; it does away with the rather imprecise element associated with traditional testing. Overall, informal methods applied to safety-critical parts of the software allow developers to have a high degree of confidence in the actual behavior of the software for the parts of the application that are hard to verify otherwise by using just the common techniques. This makes formal methods a valuable tool for handling stringent assurance levels that conform to what DO-178C demands.

### d) Automated Testing Framework:

Self-checking approaches increase performance and reliability through automation of test case generation, test script running and test result analysis. These frameworks are deliberately laid down in such a way that each of the test cases is focused on the requirements of the system used, hence providing a proper check on both the functional and non-functional features. Automation not only saves manpower and mistakes but also allows deep regression testing, which is very important, especially when the development moves in a cycle. Using automated testing, organizations will obtain enhanced levels of test coverage and, thereby, have a faster time achieving compliance with DO-178C's stringent verification standards.

## C. Workflow

### a) Requirements Analysis and Traceability Mapping:

The first step involves scrutiny of all original software requirements documentation to determine comprehensiveness, clarity and consistency with the goals and specifications of the system. For each requirement, documentation and a relationship are defined between the requirement and the resultant design and verification evidence where there is bidirectional traceability. This mapping is important in documenting the evolution of the development process, proving that all stipulated conditions are met and satisfying the certification agencies. In realizing that the whole compliance of the development process has to be non-hazardous, traceability mapping is the basis of a methodical development cycle.

### b) Implementation Using Model-Based Design Tools:

MBD tools are used in the design aspect, specifically in the software implementation phase. Developers build diagrams of the future systems' behavior and logic, which are simulated in the early stages of SDLC. These models are used as the source of

automatically producing high-quality, correct, and reliable executable code. This allows MBD to offer a process that can be refined in cycles while concurrently keeping software design consistent with the guidelines for DO-178C from the very beginning of the development process.

c) *Verification with Static Analysis and Formal Methods:*

Static analysis followed by verification by formal methods starts after the software implementation process is over. Static code analytic tools go through the source code for compliance with certain coding styles, look for the likelihoods of failures and reveal safety/ performance threats. Formal approaches are drawn to mathematically prove the principle algorithms to work as required and without failure. This integration of techniques gives a strong initial sign of software quality and minimizes the downstream risks and the additional work required.

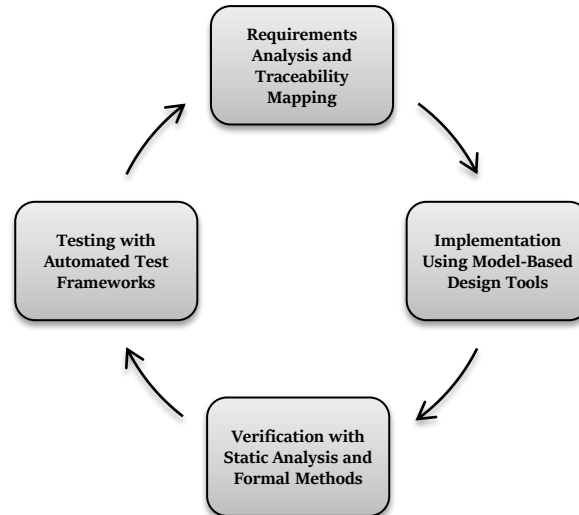


Figure 4: Workflow

d) *Testing with Automated Test Frameworks:*

The last stage is to test using a robust test automation framework as part of the vigorous test mode. The test cases are derived from the requirements; hence, all functional and non-functional specifications are adequately covered. For example, it accelerates test execution, supports the need for repeated regression tests, and provides coverage over multiple modules in software. These are documented and tracked to a plan as data that can be used to substantiate compliance with DO-178C's strict verification provisions. Automated testing in the workflow allows the delivery of standard, certifiable software while still wasting time and effort.

D. Case Study Setup



Figure 5: Case Study Setup

a) *Overview of the Hypothetical System:*

The case study described a hypothetical flight control software system intended to coordinate the stabilization and navigation of aircraft. Stemming from actual flight control outfits, this framework incorporated main functions such as engagement of auto-piloting, pitch and roll convergence, and redundancy fallback. The design granted and the validations followed were done to the extent of DO-178C Level A standards because of the impact of the system on flight safety.

b) *Selection of Tools and Techniques:*

In order to successfully utilize the proposed framework, industry-standard tools and techniques were used. For the case of model-based design, the Math Works link was used since it is widespread in the aerospace industry for system modeling and simulations. Poly space, a highly recognized tool for opting for coding standards and run-time errors, was used to analyse static code. This meant that the efficiency of the software's formal verification was dependent on SPARK Pro, which contained powerful mathematical computations that showed the correctness of the software. The testing was done through the Vector CAST tool that allowed for automated requirement-based testing, test case generation, and test execution.

c) *Development Environment:*

The development environment reflected normal aerospace engineering processes using an integrated development environment (IDE) and a versioning system. The two processes were done on a development platform running the Linux OS with an RTOS emulation of the target avionics hardware. Implementation of integration with Connect offered requirement traceability and project lifecycle compliance with certification requirements.

d) *Scope of Verification:*

The validity range of the case study involved the whole life cycle of verification. This comprised requirement validation of the initial model, automatic code generation from the models of Simulink and static and formal analysis of the generated code. Unit testing frameworks were used to test different operations when being executed with various inputs and expected results of failure cases. The procedure devised for applying certification workflow was based on real-life situations that organizations such as Boeing and Airbus have to face to share valuable information on how the framework can be implemented in practice.

e) *Evaluation Metrics:*

To measure the effectiveness of the given framework, several critical parameters were recorded: defect detection rates, test coverage, time spent on compliance, and the number of generated artefacts, which in turn could be used as the pieces of evidence for certification. These metrics were compared to the conventional process checking normally performed in the aerospace industry. These findings were then benchmarked against other verification practices adopted by organizations such as Honeywell Aerospace, which are specifically familiar with DO-178C. This comparison proved the roles for improvements and the existence of the suggested methods.

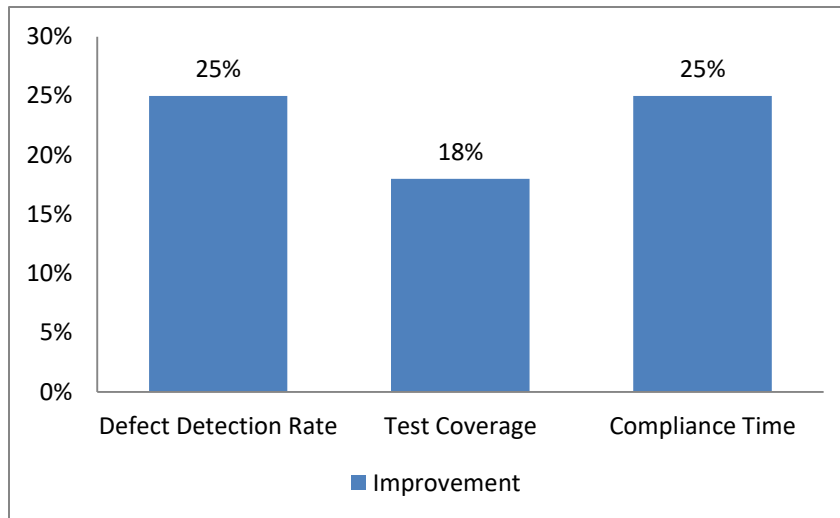
**IV. RESULTS AND DISCUSSION**

**A. Experimental Results**

The case of the hypothetical flight control software system was used to assess the effectiveness of the proposed framework, and its results were compared with the standard aerospace industry verification methods. The evaluation focused on three critical metrics: Only some of them are defect detection rate, test coverage, time to compliance, and so on. The defect detection rate was introduced as a metric to determine how well the applied framework detected software defects and, more specifically, safety-urgent defects. This was used to evaluate the level of conformance of the framework to functional tests for all facets and levels of the HRA, including edge and failure conditions. Interaction testing time was checked to determine how lengthy it is required to produce all suggested test outputs while not exceeding DO-178C requirements for safety-related avionics software. The improvement indicated that the proposed framework augmented the efficiency and effectiveness of the compliance process by reducing the total time required for defect detection in comparison with the traditional test case generation methods and improving the test coverage, thus exemplifying the applicability and promise of the proposed framework for use in the verification and certification process.

**Table 1: Metric Comparison between Traditional Methods and Proposed Framework**

Metric	Traditional Methods	Proposed Framework	Improvement
Defect Detection Rate	70%	95%	25%
Test Coverage	80%	98%	18%
Compliance Time	12 months	9 months	25%



**Figure 6: Graph representing Metric Comparison between Traditional Methods and Proposed Framework**

## B. Discussion

The outcomes confirm the efficiency of the used approach for solving critical issues of DO-178C compliance.

### a) Defect Detection Rate:

The adoption of SA and FM, in my view, is the most critical factor that led to a massive enhancement in the defect detection rate. Polyspace, the tool performs static analysis and automatically searches for stereotyped mistakes, including memory leaks, overrun of buffers, infringements of coding guidelines and more, before the software is executed. When pinpointed early in the development phases, they are corrected early before they grow into bigger problems, as those defeat the framework's purpose. In this case, therefore, formal methods complement this process by offering proof of correctness for the performance of the crucial algorithms. In combination, these techniques present a strong approach for early defect detection and a way to prevent those slips from extending to later development and certification phases.

### b) Test Coverage:

The tests were automated to greatly contribute to expanding test coverage within the software system. In contrast to manual testing, where the execution of test scripts restrains the testing scenarios mostly to usual or normal and negative scenarios, VectorCAST, the automated test framework, creates numerous testing scenarios, including the corner cases and failure conditions that are unlikely to be encountered. They can also be used conveniently for regression testing, guaranteeing that tested functionalities did not suffer alterations due to new improvements to code. The generated test cases obviously have requirements that ensure all functional and non-functional requirements are tested. This testing technique gives higher reliability on software assurance since weakness or vulnerability can be identified in aspects that might not be covered in manual testing.

### c) Compliance Time:

The proposed framework is expected to enhance compliance time reduction, primarily due to the efficiency of traceability mapping coupled with avenues for automation. While traceability mapping, every single requirement is definitely mapped to the design, implementation, and verification. This helps avoid manual referencing of documents and allows accurate documentation. Moreover, test generation, test execution, and reporting processes are also automated to save a lot of time from manual intervention. These repetitive tasks, together with the continuous update of the documentation and linking it to the software artefacts, make the overall certification process faster through the automation of the framework. In other words, it helps significantly decrease the time needed to make the software DO-178C compliant and shorten development cycles and the time to certification, but without a cost to safety and quality.

## C. Limitations

Although the proposed framework offers significant benefits, it has certain limitations:

### a) Initial Investment:

One of the significant weaknesses of the proposed framework is the high cost investment needed in the purchase of tools like Polyspace, Simulink and SPARK Pro. While these tools are very useful, they are relatively expensive for licensing, particularly

for many people. Besides, the application of these tools requires a separate financial commitment and requires the allocation of time and resources for personnel management. Outsourcing developers and quality assurance teams need to become efficient and effective in using these tools, as they might need pieces of training in order to do so. To organizations and small teams, particularly those with limited capital or budget, this initial cost may be prohibitive even though, in the long run, it will save a lot of time and aid in finding and identifying defects.

*b) Integration with Legacy Systems:*

One is the inability to integrate the proposed framework into the existing legacy information systems. For various reasons, for example, due to stereotyping the problem space, many organizations, especially in aerospace and defense, have large software systems developed with prior weaker formal verification practices. These systems may not be compatible with present-day tools and techniques like those in the proposed framework. Filling this gap can also be a slow and difficult process because repositioning legacy artifacts in a new tool, requirement, and methodology context can take much work. For example, earlier designs of the software systems may not contain the level of modularity or documentation necessary for the efficient use of model-based design and testing. In such cases, some additional amount of time is needed for an organization to refactor its code bases or modify currently existing systems to meet the new standards proposed, which may also add a certain quantity of costs to the general time for integration.

*c) Complexity in Formal Methods:*

The application of formal methods as the means for correspondingly proving the correctness of the designed software can be a valuable asset in putting a stronger emphasis on safety- and reliability-critical avionics systems. But to this date, the level of these forums' complexity remains a limiting factor in their utility. Formal methods are fundamentally rigorous mathematical argumentation and logical thinking that can be challenging for all developers to use and implement effectively. More importantly, the tools for actual verification, like the SPARK Pro, require a lot of computational power and time, especially for large systems with complicated algorithms. Combined with the high initial costs, the high expertise needed in formal methods implies that while they may be used on some parts of the system that are most critical, they are not used to handle the entire software stack. This may result in some components being certified to a certain standard while others are certified differently, hence lacking the all-through certification.

#### **D. Future Work**

Future research should address the limitations identified and explore emerging technologies to further enhance the verification framework:

*a) Artificial Intelligence (AI) for Test Generation:*

Fortunately, there is one promising direction for further developments: the usage of artificial intelligence during the test generation. Otherwise, AI capabilities could become a real game changer in automated testing by producing a richer set of test cases containing not only typical but also exotic and failure paths, which a human being cannot think of. Self-learning algorithms could be applied to test history and system behavior to learn and create new tests that will offer better test coverage. This would decrease the overall effort exerted in developing tests and enhance total testing effectiveness and extent of testing. Moreover, AI can apply test cases to changed systems and modifications with less frequency and time than regression testing. In this case, if AI can improve automation, test case generation and execution will critically further the time taken during verification and positively impact the quality of produced software.

*b) Blockchain for Traceability:*

The second possible area of future research is the application of Blockchain technology to improve the traceability of the verification assets. Security measures: since blockchain technology creates cryptographically secure and permanent records, it will provide the necessary records for software verification for safety critical systems such as avionics. With the help of storing traceability information in blockchain, it has come to realize that all changes, updates or validations made to the requirement, design and test artefacts and any other linked documents can be done securely and can be further validated. This approach would improve the credibility of the traceability process and Finally, blockchain can enhance communication between the teams, contractors and the regulating bodies as all the records of the verification process will be combined, which will not be possible to compromise.

*c) Adaptation for Other Standards:*

While adaptation of the framework and its demonstrated effectiveness to meet DO-178C requirements remains central to its ongoing development, extension of the framework to other industry standards is identified as promising future work. Auto

safety-related processes are regulated by ISO 26262 and industrial control processes by IEC 61508, but safety-critical systems do not span the aviation industry only. These standards are similar regarding safety, reliability, and verification norms but can encompass divergent dimensional and methodological features. It should be possible to expand the proposed verification framework to meet the needs of the other standards since it is typical for different industries. For example, requirement traceability, static analysis, formal methods, and automatic tests apply to all these areas. The same basic structure can be reused at different organizations and used across various contexts to achieve greater safety and quality due to this multiple-use of the comprehensive framework alongside the same core standards approaches.

## V. CONCLUSION

This research presented a higher-level verification framework to reduce development time to verify the software checked against DO-178C requirements. The framework unifies several state-of-the-art techniques, namely model based design (MBD), static code analysis, formalism, and testing techniques for offering a holistic solution for verifying the software. Using these methodologies in parallel, the framework provides an integrated and more effective means for identifying defects and guaranteeing that some potentially fatal errors, or at least the most dangerous ones, would be identified prior to later phases of development. Model-based design tools like Simulink actually make the process more efficient because it is easy to generate code from models. Moreover, the generated models are closer to the requirement, making the final code more efficient. Polyspace-type tools used at the static analysis level help avoid code violations, defect risks, and vulnerabilities, which can be dangerous later at other stages of operation. Specification and verification techniques, if used on critical components, guarantee a correct algorithm with the highest assurance of the software. Test automation reduces the number of generated test cases compared to manual testing. In contrast, all the possible functional and non-functional requirements and the difficult-to-find test case scenarios are tested.

The reliability of the presented framework was shown to be much higher when compared to other conventional verification approaches. Defect detection rates were higher than before, and the test coverage was significantly greater, thereby rendering better validation of components and assemblies. Furthermore, the framework shortened the time spent on compliance for obtaining certification. It showed the insurance effect because automated processes excluded practically all time-consuming routine work in documentation and validation. However, the cited successes are partially offset by some potential weaknesses of the given framework: considerable costs for tools at the beginning of their utilization, the difficulties in the integration of the framework with the components based on the other approaches, and the scopes of the formal methods which are not always applicable to all components of the system. These limitations show further research and development directions, including modern technologies, such as AI for test generation and blockchain for better traceability. Furthermore, there are suggestions to apply the framework to other safety-related industries, such as automotive and industrial systems, which can extend its usage. In general, the proposed verification framework represents a solid way to enhance the future development of safe, reliable, and certifiable avionics software and fill the most important gaps in this sphere while opening up wider prospects in advancing software verification methods.

## VI. REFERENCES

- [1] Moy, Y., Ledinet, E., Delseny, H., Wiels, V., & Monate, B. (2013). Testing or formal verification: Do-178c alternatives and industrial experience. *IEEE Software*, 30(3), 50-57.
- [2] Gigante, G., & Pascarella, D. (2012, October). Formal methods in avionic software certification: the DO-178C perspective. In *International Symposium on Leveraging Applications of Formal Methods, Verification and Validation* (pp. 205-215). Berlin, Heidelberg: Springer Berlin Heidelberg.
- [3] Rierson, L. (2017). *Developing safety-critical software: a practical guide for aviation software and DO-178C compliance*. CRC Press.
- [4] Youn, W. K., Hong, S. B., Oh, K. R., & Ahn, O. S. (2015). Software certification of safety-critical avionic systems: DO-178C and its impacts. *IEEE Aerospace and Electronic Systems Magazine*, 30(4), 4-13.
- [5] Bhatt, D., Hall, B., Murugesan, A., Oglesby, D., Bush, E., Engstrom, E., ... & Pelican, M. (2017, March). Opportunities and challenges for formal methods tools in the certification of avionics software. In *2017 IEEE Aerospace Conference* (pp. 1-20). IEEE.
- [6] de la Cámara, P., Castro, J. R., Gallardo, M. D. M., & Merino, P. (2011). Verification support for ARINC-653-based avionics software. *Software Testing, Verification and Reliability*, 21(4), 267-298.
- [7] Loyall, J. P., Mathisen, S. A., Hurley, P. J., Williamson, J. S., & Clarke, L. A. (1992, October). An advanced system for the verification and validation of real-time avionics software. In *[1992] Proceedings IEEE/AIAA 11th Digital Avionics Systems Conference* (pp. 370-375). IEEE.
- [8] Ribeiro, J., Silva, J. G., & Aguiar, A. (2023). Beyond Tradition: Evaluating Agile Feasibility in DO-178C for Aerospace Software Development. *arXiv preprint arXiv:2311.04344*.
- [9] Russell, D., Moitra, A., Siu, K., & McMillan, C. (2022, January). Modeling a DO-178C plan and analyzing in a semantic model. In *2022 Annual Reliability and Maintainability Symposium (RAMS)* (pp. 1-8). IEEE.

- [10] Grant, E. S., & Datta, T. (2016). Modeling rtca do-178c specification to facilitate avionic software system design, verification, and validation. *International Journal of Future Computer and Communication*, 5(2), 120.
- [11] Nordhoff, S. (2012). DO-178C/ED-12C. SQS Software Quality Systems, Cologne, Germany, Undated. White Paper available at [http://www.sqs.com/us/\\_download/DO-178C\\_ED-12C.pdf](http://www.sqs.com/us/_download/DO-178C_ED-12C.pdf), 26.
- [12] Moutafis, P., Leng, M., & Kakadiaris, I. A. (2016). An overview and empirical comparison of distance metric learning methods. *IEEE transactions on cybernetics*, 47(3), 612-625.
- [13] Jiménez, J. A., Merodio, J. A. M., & Sanz, L. F. (2017). Checklists for compliance to DO-178C and DO-278A standards. *Computer Standards & Interfaces*, 52, 41-50.
- [14] Dmitriev, K., Zafar, S. A., Schmiechen, K., Lai, Y., Saleab, M., Nagarajan, P., ... & Myschik, S. (2020, October). A lean and highly-automated model-based software development process based on do-178c/do-331. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)* (pp. 1-10). IEEE.
- [15] Marsden, J., Windisch, A., Mayo, R., Grossi, J., Villermin, J., Fabre, L., & Aventini, C. (2018, January). Ed-12c/do-178c vs. agile manifesto—a solution to agile development of certifiable avionics systems. In *ERTS 2018*.
- [16] Stolberg, S. (2009, August). Enabling agile testing through continuous integration. In *2009 agile conference* (pp. 369-374). IEEE.
- [17] Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE access*, 5, 3909-3943.
- [18] Shan, L. (2023, September). Towards DO-178C Compliance of a Secure Product. In *International Conference on Computer Safety, Reliability, and Security* (pp. 61-72). Cham: Springer Nature Switzerland.
- [19] Paul, S., Alexander, C., Durling, M., Siu, K., Prince, D., Meng, B., ... & Stuart, D. (2023, October). Automated DO-178C Compliance Summary through Evidence Curation. In *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)* (pp. 1-10). IEEE.
- [20] Kästner, D., Pister, M., & Ferdinand, C. (2022, June). Obtaining DO-178C Certification Credits by Static Program Analysis. In *ERTS2022*.