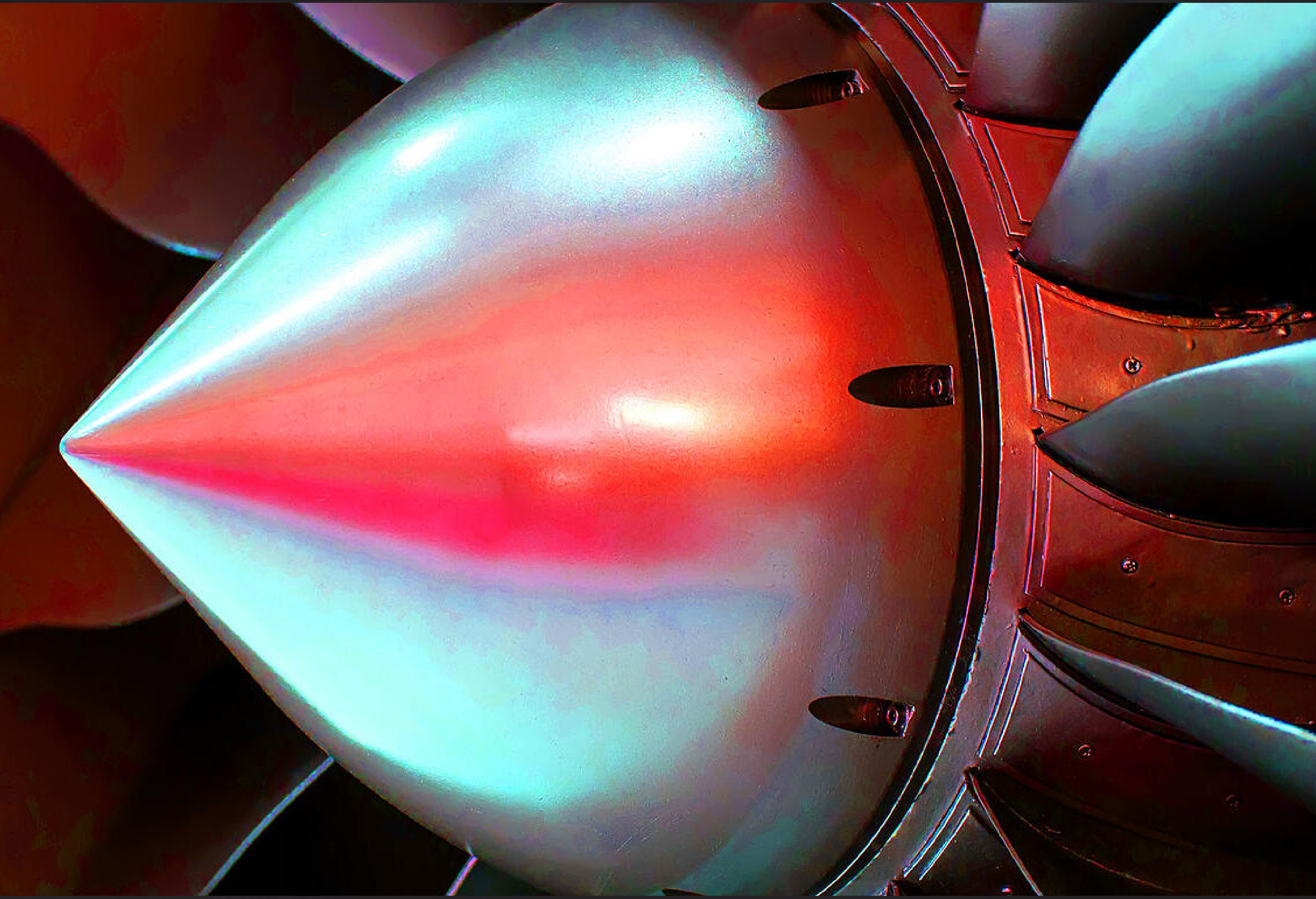
GRAMMATECH

DEVSECOPS IN SAFETY CRITICAL AVIONICS SOFTWARE AND THE ROLE OF STATIC ANALYSIS



TRUSTED LEADERS OF SOFTWARE ASSURANCE AND ADVANCED CYBER-SECURITY SOLUTIONS

WWW.GRAMMATECH.COM

INTRODUCTION

DO-178C, Software Considerations in Airborne Systems and Equipment Certification, is a standard published by RTCA, Inc and developed jointly with EUROCAE, the European Organization for Civil Aviation Equipment. Alongside DO-178C is D-326A (U.S.) and ED-202A (Europe) titled "Airworthiness Security Process Specification" and is the only Acceptable Means of Compliance (AMC) by FAA & EASA for aviation cyber-security airworthiness certification, as of 2019. Officially, DO-326A is part of a set of documents (DO-326A/ED-202A, ED-205, DO-356A/ED-203A and DO-355/ED-204,) that provide

> *"guidance for aircraft certification to handle the threat of intentional unauthorized electronic interaction to aircraft safety. It adds data requirements and compliance objectives, as organized by generic activities for aircraft development and certification, to handle the threat of unauthorized interaction to aircraft safety"*

In simple terms, modern avionics systems requiring certification by the FAA or EASA, must satisfy the requirements of both DO-178C for safety considerations and DO-326A (set) for security.

DEVSECOPS

DevOps has sprouted from the combination of software development and systems operations to make sure that software development is not done in a vacuum, but in combination with the teams that operate these systems in the real-world.

The next step in this improvement of software development methods is DevSecOps, where Security is included as a critical part of the development process. The realization here is that a security failure is the same, or worse, as a quality failure. Defects in fielded product impact the bottom line as well as company reputation. It is even worse, if a review after the fact determines that these defects could have easily be avoided. So DevSecOps is the integration at the team level of the teams building the software, operating the software and securing the software.

In addition, a key reason to build security into agile and continuous processes is to build upon the knowledge that accumulates over the project. It's not reasonable to expect software teams to understand their complete attack surface, for example, at the beginning of the project. Building security into day-to-day operations accumulates expertise and knowledge. Starting early is key. With that in mind, DevSecOps is often illustrated as follows on the DevOps flowchart – security in every part of cycle:

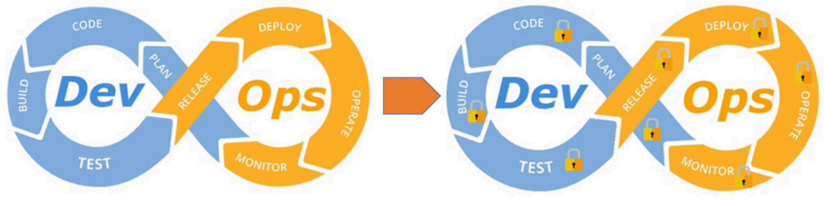


Figure 1: The DevOps cycle, continuous code, build, test and deploy. DevSecOps is integrating security into all parts of the cycle. Source: "Tripwire - Security at the Speed of DevOps"

DEVOPS AND DEVSECOPS IN SAFETY CRITICAL SOFTWARE

A key aspect of adopting DevOps (and DevSecOps) is automating tedious, potentially error-prone, manual steps in the development process. Part of this is the move to continuous integration (CI) and deployment (CD). The aim of each is to speed up the time to quality and security by finding and fixing integration and deployment issues earlier in the lifecycle. A continuous process provides feedback in terms of defects and vulnerabilities (in addition to safety, performance, and other quality attributes) that improves subsequent iterations. This approach has proven successful in IT software and is gaining ground in embedded development and safety and security critical software.

The biggest challenge to adopting Agile processes, Dev(Sec)Ops and CI/CD in safety critical software is the need to satisfy audits, validation and certification requirements. Many Agile methods and DevOps processes and techniques aren't defined in detail, letting teams adopt the details as they see fit. However, safety and security standards require a rigorous, well-defined process. This is not incompatible but it does require software teams to define and document their DevOps tools, processes and techniques well and in detail. An important example of this is traceability. Proving requirements are satisfied with proven validation evidence is important for proving system functionality and airworthiness. Any DevSecOps process would have to manage traceability precisely.

HYBRID APPROACHES TO AGILE, CI/CD AND DEV(SEC) OPS IN THE V-MODEL OF SOFTWARE DEVELOPMENT

The traditional software development approach, often categorized as the "waterfall" method implies that each major stage of the software development life cycle is complete from start to finish before the next phase starts. In safety critical systems this often characterized by the V-model, as shown below:

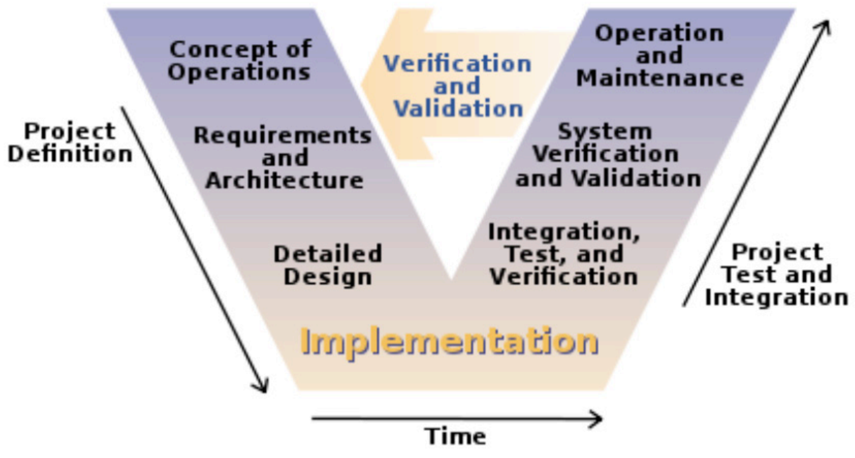

Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

Figure 2:The "V model" of software development showing the typical timeline of a software development project.

The V-model is important for safety critical software since it defines the dependencies of each phase and the traceability links needed from phase to phase (e.g. requirements drive system architecture, system testing validates requirements, etc.) This is seen as the antithesis of Agile but adopting a hybrid approach that leverages continuous processes such as CI/CD and Dev(Sec)Ops is still possible. Although the timeline flows towards more complete implementation, verification and validation, it's still possible to iterate within major phases of development. The V-model might imply waterfall development but the standards do not dictate any process and certainly do not prevent Agile and iterative techniques. Most software development teams building software with high safety requirements have by now incorporated iterative and agile techniques in their process.

Define and plan with the waterfall/V model: In practice many avionics systems have requirements defined up front as part of request for proposal (RFP) process during vendor selection. It is also probable that milestones are established as part of large-scale airframe project where deliverables are well defined. In such cases, planning around these requirements and milestone is necessary. These requirements feed the design and implantation phases that can still be iterative, Agile processes.

Agile methods for software development and verification: The software development method chosen during the design, implementation and testing of code is left up to the manufacturer as long as it meets the basic criteria of good engineering practices with traceability, safe and secure practices, and reporting and documentation evidence for results (to name a few.) Agile and iterative methods can work well in this phase despite the entire lifecycle not necessarily working within the Agile framework. In fact, the approach leads to better results by shifting many important parts of development earlier such as testing.

CI/CD and DevOps to reduce integration issues: Most developers would agree the biggest issue with waterfall development is the issues that arise during integration. Often called "Big Bang integration" many undetected issues raise their head during this phase and it results in rework and "crunch" time to fix bugs. One of the key benefits of CI/CD is reducing these Big Bang issues by integrating software much earlier (and continuously) than is typical in the waterfall process. In conjunction with an iterative/Agile process, this approach required developers to create incremental functionality is deployed and tested thoroughly before moving onto the next "chunk." Going further with a full DevOps approach, many of the issues in the entire software deployment and release chain is eased by attacking them earlier, in other words "shift left" on the software development timeline. Although CI/CD and DevOps is more challenging to adopt in embedded safety critical systems, many teams are using this approach already. More and more projects in aviation as well as in large scale projects in DoD are getting closer to full CI/CD and DevOps, including delivery to the end-customer for early validation of subsets of requirements, but also for cyber security testing. Now, these deliveries seldom include a full flyable air-frame, but do certainly include integrated sub-systems that can be tested on test benches.

DEVSECOPS AND DO-326A/DO-356

DO-326A defines an air-worthiness security process (AWSP) which at a top level define certification, security risk assessment and security development activities. Security risks evaluated during assessment activities require security development activities to mitigate the risk to the aircraft. These activities are meant to be integrated into the safety processes required for the software. The details of the methods and tools for this security process are defined in DO-356, “Airworthiness Security Methods and Considerations”. These standards encompass a large scope, much more than the discussion here. However, as it applies to new or existing code development and maintenance there are good arguments for adopting DevSecOps in these applications.

SECURITY IN CODE

Good engineering practices dictate adoption of coding guidelines or standards such as MISRA, or SEI CERT guidelines. This approach assures newly developed code follows industry best practices. However, a coding standard by itself does not prevent all complex security vulnerabilities (nor is it practical to implement coding standards on existing code.) For example, consider a vulnerability caused by a buffer-overrun-write, with a root cause that follows a complex path where a variable is tainted (contains data intended to exploit) through input from the outside of the system. To find these problems in the code more sophisticated static analysis tools are needed that include dataflow analysis and abstract execution.

SECURITY AS CODE

An interesting approach that has come out of DevSecOps practice is the concept of treating security requirements in the same way as safety and functional requirements. (This is, of course, required in modern avionics software.) Guided by the outcome of detailed threat analysis and the implementation of security controls, then to validation through testing, and of course documentation. This is the key way to integrate security into DevOps and a good way to build security into the development culture and have software teams communicate using a familiar language. Certainly, security implementations that end up as actual code with associated tests can be automated in the same fashion as other code. Automation tools apply here as would any tool that works with requirements, tests, documentation, etc.

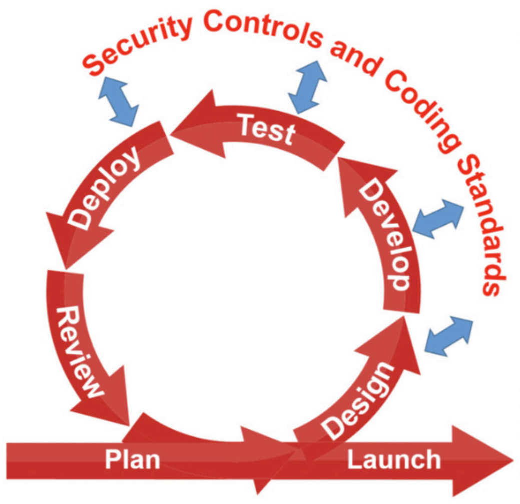


Figure 3: DevSecOps requires security requirements, controls and coding standards fed into each part of the pipeline. Importantly, feedback is required to close the loop.

ROLE OF STATIC ANALYSIS IN DEVSECOPS IN SAFETY CRITICAL SOFTWARE

Static analysis tools are designed to integrate well with just about any software automation tool chain and development methodology and process. This is mainly due to the fact they can be used locally by developers at their desktop for instantaneous feedback and used to analyze a complete build whether that's done hourly as part of a CI/CD process, or whenever. In addition, these tools are completely autonomous since they require no interaction with testers or developers. They are applicable whenever it makes sense to check for bugs and security vulnerabilities in the code:

- **Develop:** This is the critical time to detect any new security vulnerabilities, as soon as developers write new code (or test cases) even before it's submitted to a build or software control system. GrammaTech CodeSonar, for example, presents these vulnerabilities immediately in the developer's IDE just like a compiler warning, providing easy and actionable corrective action (such as vulnerability assessment, root causes and control and data flow traces). Turnaround time is important here, feedback needs to be fast, such that the developer is not bogged down waiting for builds to finish. The integration into the IDE is important here as it lessens the cognitive load on the software developer. He does not need to leave his favorite development environment and can review and assess the static analysis output right there in his IDE. These tools are essential in this phase for ensuring secure coding standards are met. Violations of the coding standard and potentially dangerous coding practices are flagged for developers to investigate.

- **Test:** This is where all the changes from all developers are brought together for more comprehensive testing, static analysis plays an important part in that process.

- **Deploy**: Static analysis tools analyze deployable binaries and libraries. CodeSonar, for example, supports binary code analysis for C/C++ code that detects the same types of vulnerabilities as source analysis. This is a good practice to detect bugs introduced during building and deployment of deliverables.

- **Review**: Static analysis reporting and analysis that software teams can use to evaluate individual warnings but also higher-level assessments of application quality and security. Reports from SAST should be part of the cycle assessment and planning for each cycle.

These integrations into the DevSecOps cycle are illustrated below:

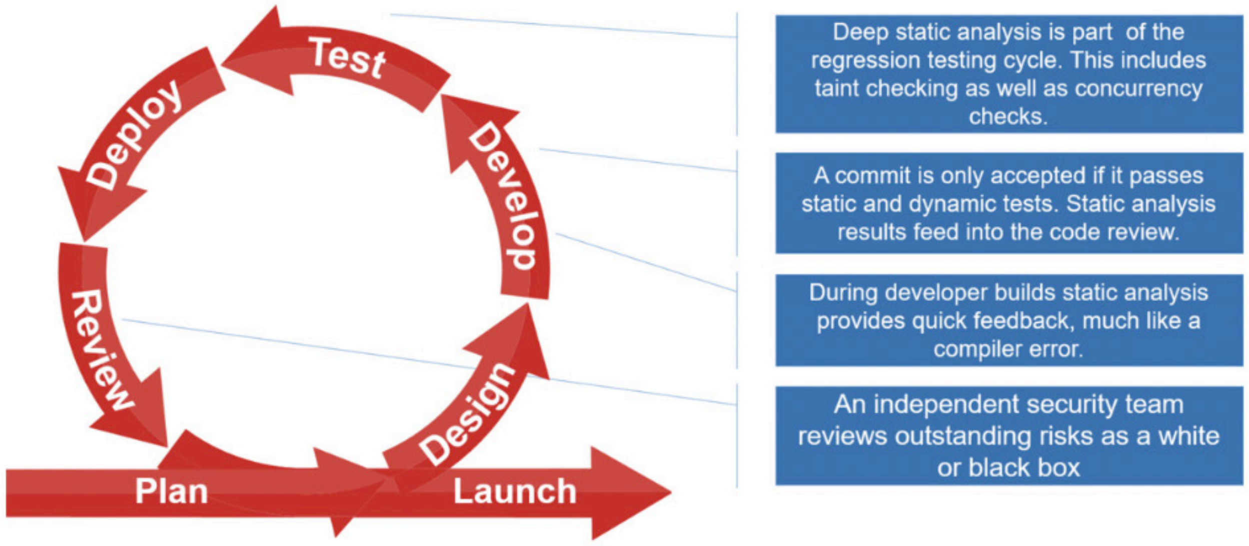

Test
Deploy
Develop
Review
Design
Plan
Launch
Deep static analysis is part of the regression testing cycle. This includes taint checking as well as concurrency checks.
A commit is only accepted if it passes static and dynamic tests. Static analysis results feed into the code review.
During developer builds static analysis provides quick feedback, much like a compiler error.
An independent security team reviews outstanding risks as a white or black box

Figure 4: The role of Static Analysis in DevSecOps

THIRD-PARTY SOFTWARE

Use of third party code such as commercial off-the-shelf software (COTS) and open source software is a fact of life in embedded software development and is increasingly part of avionics software. Static analysis tools can analyze third party source and binaries to discover defects and security vulnerabilities in software that could be impossible to test otherwise (without including it and running it, an expensive option).

If source code is not readily available, this does not preclude the ability to detect bugs and security vulnerabilities. In addition, security teams can use binary analysis (available in CodeSonar for Binaries) to perform “black box” analysis of product deliverables.

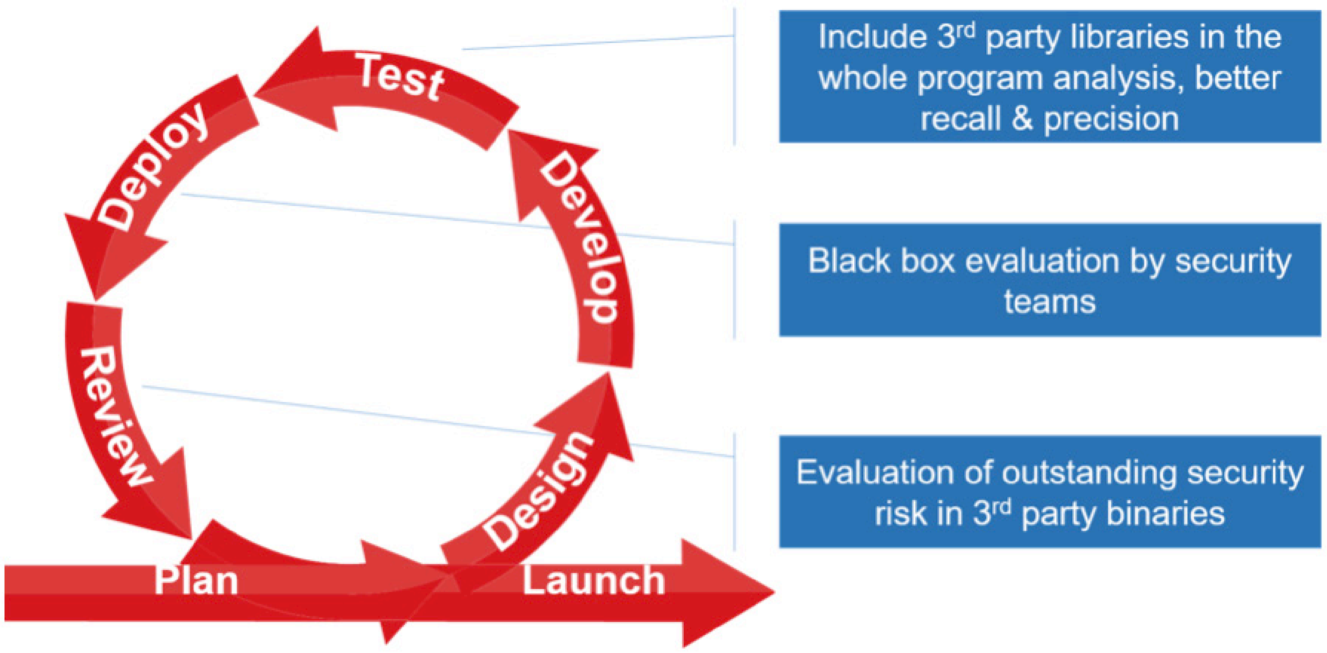

Test
Deploy
Develop
Review
Design
Plan
Launch
Include 3rd party libraries in the whole program analysis, better recall & precision
Black box evaluation by security teams
Evaluation of outstanding security risk in 3rd party binaries

Figure 5: The added benefit of binary analysis in a continuous integration process.

TOOL QUALIFICATION

Although software development tools used in Avionics aren't certified in and of themselves. They do require qualification to certification bodies such as the FAA. As such, it's important that any tools used have an acceptable pedigree and the ability to meet tool qualification requirements. For example, GrammaTech CodeSonar is certified for IEC 61508, EN50128, and ISO 26262 standards, which means that an independent certification body, Exida in this case, has analyzed the development process used to build CodeSonar and certified that it satisfies the requirements to be used in developing safety-critical software.

The process to qualify tools for use in software certified to DO-178C is described in DO-330. In short, this document describes the requirements that a tool needs to meet based on how the tool is used in the process. The requirements for a code-coverage tool are different compared to the requirements for a compiler, which are different from the requirements for a static analysis tool. A static analysis tool is typically considered a criteria 3 tool (based on section 12.2), which is a tool that, within the scope of its intended use, could fail to detect an error. Within these limits, the standard then defines how to ensure that the tool behaves as expected in the software development process. At a high level, this means that the project needs to describe how it plans to use the tool in a Tool Operational Requirements Document, it needs to provide a Tool Qualification Plan that describes how to proof that the tool indeed does perform correctly. The latter typically require a set of test artefacts and expected results.

GrammaTech and its customers have a lot of experience in assisting customers with projects that require certification and have the artefacts available to assist with tool qualification.

SUMMARY

Modern safety critical avionics software requires rigorous security engineering that interleaves with the established safety practices in place. New cyber security standards are requiring security design from the very beginning of product lifecycle. To address these requirements, there are advantages to modern software development practices found in Agile, continuous integration and deployment and DevSecOps practices. Despite many airframe projects being contrained at the high level to a “V model” or waterfall approach, hybrid methods mixing V model and continuous process together are a good alternative.

Focusing on the “shift left” of security and safety verification and assessment reduces the effects of “big bang” integration that plagues the V model approach. Just as it is impossible to build-in safety into software, the same holds true for security. Security must be part of the product concept and built-in.