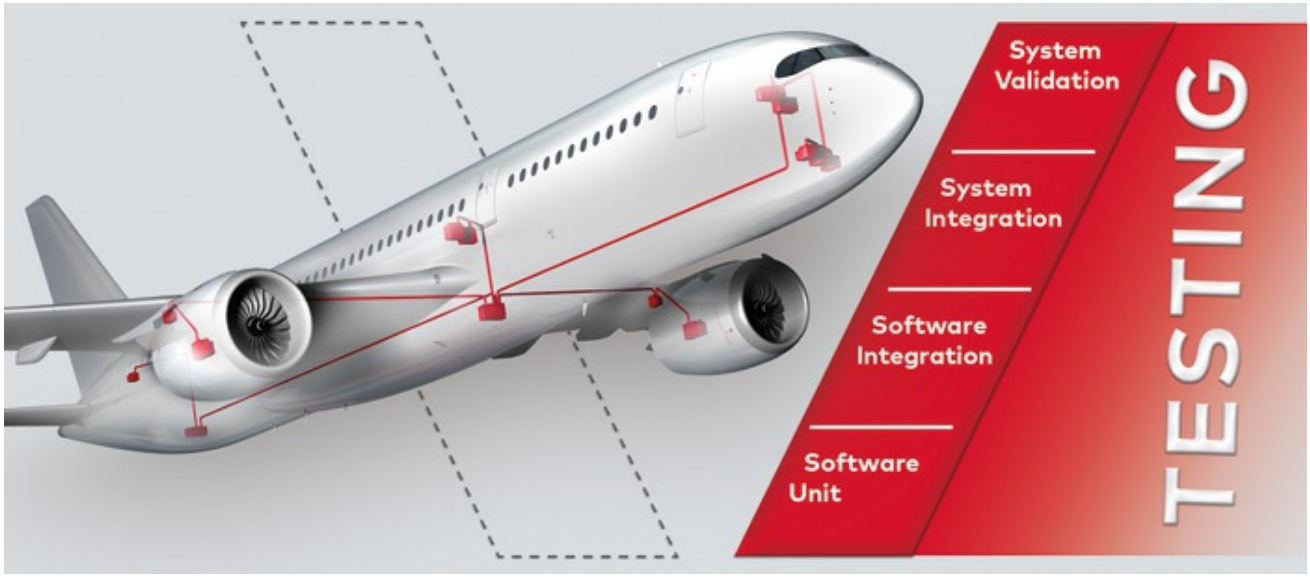
CAN Newsletter Online

CAN NEWSLETTER MAGAZINE

# Best in test

**Due to the parallels between systems from the aerospace and automotive worlds, it is possible to transfer proven concepts and processes from the automotive industry to avionics. Vector describes it's approaches and concepts.**




*(Source: Vector Informatik)*

*The complete article from Vector Informatik is published in the March issue of the CAN Newsletter magazine 2020. This is just an excerpt.*

Continued development of comprehensive and structured testing methods and tools for electronically net- worked aircraft and cabin systems is not only necessary for economic reasons. With the recent safety-critical pilot assistance systems such as Enhanced Vision and Run- way Overrun Protection Systems or with "wireless" cabin functions each requires appropriate testing strategies that are compliant to the regulatory rigors assigned to them (i.e. DO-178C). This article describes Vector's approaches and concepts.

The software in avionics and ground-based systems are bounded by strong regulatory standards DO-178C and DO-278 respectively. With failure regarded as "not an option", significant analysis and effort is put into the verification and validation of these systems. In fact, industry wide, in a typical project fifty percent of the development budget is used for structural testing the software according to Federal Aviation Administration (FAA) DO-178C Level A [1]. The ability to automate and simulate these systems can greatly assist in reducing the overall effort, and hence the implementation costs.

There are three major phases of verification and validation in avionics and ground-based software (Figure 1): unit testing, integration testing, and system/functional test- ing. In each phase, test cases need to be derived from their appropriate level of requirements with full traceability between both.

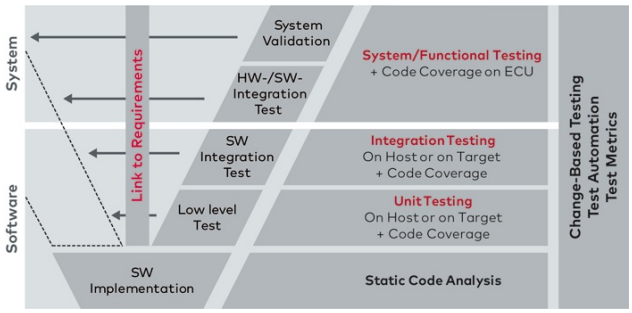



*Figure 1: The major phases of verification and validation in avionics and ground-based software (Source: Vector Informatik)*

While the concepts and methodologies for low level testing have been reasonably consistent over the years, the introduction of more networked systems based on the CAN and AFDX protocol, and the drive for code reuse, demands innovations in the approach as to how the software should be tested. To find good solutions, other industries can be considered that have successfully deployed complex net- worked systems, with rapid time to market demands and highly critical functionality. An example is the automotive market, with its drive by wire systems, autonomous vehicle technology, 18 month to 24 month development cycle and CAN/Ethernet networked platforms.

The similarities in particular in CAN-based systems make it possible to transfer proven concepts and processes from the automotive industry into the avionics domain. CAN is currently used in modern civil aircrafts like A350 and Boeing 787 for systems such as environmental control, doors, galleys, smoke detection, potable water, and de-icing. Furthermore young companies acting in the emerging market of hybrid and full electric air vehicles for new urban air mobility concepts rely on CAN-networks as well.

Due to the specific challenges like long cables, extreme environmental conditions, stringent lightning protection requirements, and long service life, adequate test strategies at all test levels must be foreseen. The approaches can be considered at three levels as described in section 6.4.3 of the DO-178C standard: low level testing, software integration testing, and hardware/ software integration testing. Finally, it is worth considering how these can be coupled into a process which provides greater agility as well as introducing shift-left strategies into the development process.

**Low-level testing**

This testing level is used to test the low-level requirements and is usually accomplished with a series of unit tests that allow the isolation of a single unit of source code. To test a single unit in isolation, a huge amount of framework code such as test drivers and stubs for dependencies (Figure 2) must be generated. Ideally, this should be done automatically with a tool that offers an intuitive and simple approach for defining test scenarios. This meets the main requirements of section 6.4.2 “requirements-based test selection” and the sub-sections “normal range test cases” and “robustness test cases” of the DO-178C standard. With the growing need for code reuse, it is very likely the same unit of source code might be used in several configurations. Therefore, it is important that the definition of a test case is not tightly coupled to the code and provides flexibility in how they can be maintained as the software evolves over time. Typically, the use of a data driven interface for the definition of test cases has proven to be more maintainable over time than a source code definition.

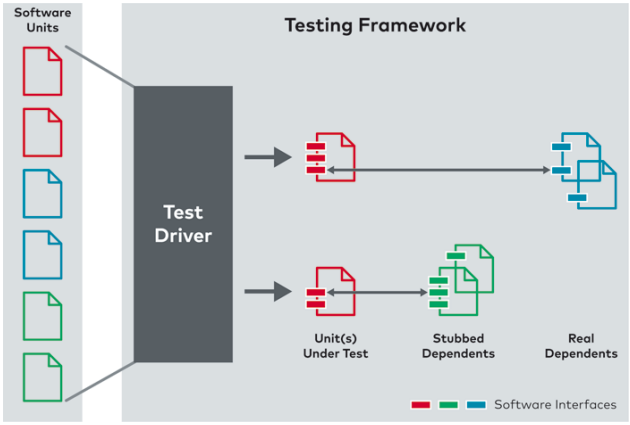



*Figure 2: Low level testing framework to test a single software unit in isolation – using framework code such as test drivers and stubs for dependencies (Source: Vector Informatik)*

This approach also means that when the source code and associated test cases are deployed in a continuous delivery workflow, as changes are made to the code, the testing framework can quickly be regenerated and the test cases appropriately remapped. Where significant changes have been made, these can be flagged for further review without breaking the rest of the automated workflow.

A good example of this is the embedded software testing platform Vectorcast, that automates testing activities across the software development lifecycle. It fully supports testing on target or using the target simulator normally provided by the compiler vendor. Structural cover- age from testing isolated components can be combined with the coverage gathered during full integration testing to present an aggregated view of coverage metrics. Vectorcast test cases are maintained independent of the source code for a data-driven test approach. This technique allows tests to be run on host, simulator, or directly on the embedded target in a completely automated fashion.

*If you would like to read the complete article you can download it free-of-charge or you download the entire magazine.*

*cw*