SOFTWARE TESTING

BEST IN TEST

Comprehensive and structured testing of electronically networked aircraft and cabin systems is necessary not just for economic reasons, but also increasingly in terms of meeting strict regulations

// ARNE BREHMER, HANS QUECKE AND NIROSHAN RAJADURAI

"IT IS IMPORTANT THAT A TEST CASE IS NOT TIGHTLY COUPLED TO THE CODE"

1

The software in avionics and ground-based systems is regulated by the standards DO-178C and DO-278, respectively. With failure being out of the question, significant analysis and effort is put into the verification and validation of these systems. In fact, industry-wide, in a typical project 50% of the development budget is used for testing software to FAA DO-178C Level A. The ability to automate and simulate these systems can greatly assist in reducing the overall effort, and hence implementation costs.

There are three major phases of verification and validation in avionics and ground-based software (Figure 2). They are unit testing, integration testing and system/functional testing. In each phase, test cases need to be derived from their appropriate level of requirements with full traceability between both.

Low-level testing is used to test low-level requirements and is usually accomplished with a series of unit tests that allow the isolation of a single unit of source code. While the concepts and methodologies for this type of testing have been reasonably consistent over the years, the introduction of more networked systems based on the AFDX (Avionics Full-Duplex Switched Ethernet) protocol, and the drive for code reuse, demand innovations in the approach to testing software. Finding good solutions means looking at other industries that have successfully deployed complex networked systems, with rapid time to market demands and highly critical functionality. One such example is the automotive market, with its drive-by-wire systems, autonomous vehicle technology, 18- to 24-month development cycle and CAN/Ethernet networked platforms.

1 // Interlinked software running different parts of an aircraft must meet the DO-178C and DO-278 regulated standards

The similarities in these systems make it possible to transfer proven concepts and processes from the automotive industry into the avionics domain. The approaches can be considered at three levels, as described in Section 6.4.3 of DO-178C: low level testing, software integration testing and hardware/software integration testing. Finally, it is worth considering how these can be coupled into a process that provides greater agility and introduces shift-left strategies into the development process, which broadly means to test earlier.

LOW-LEVEL TESTING

To test a single unit in isolation, a huge amount of framework code such as test drivers and stubs for dependencies (Figure 3) 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 subsections Normal Range Test Cases and Robustness Test Cases of DO-178C. With the growing need for code reuse, it is very likely that a certain section of source code will 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 the code can be maintained as the software evolves. Typically the use of a data-driven interface for the definition of test cases is more maintainable over time than a source code definition.

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, they can be flagged for further review without breaking the rest of the workflow.

A good example of this is the Unit Test Automation tool VectorCAST. The tool fully supports testing on targets or using the target simulator normally provided by the compiler vendor. Structural coverage 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 independently 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.

SOFTWARE INTEGRATION TESTING

This concept verifies the interrelationship of components and is also known as software-in-the-loop (SIL) testing. The aim is to bring the software components together and test them without any of the complexities of the underlying hardware. A critical aspect of testing software during this phase is the ability to simulate dependencies and interfaces in the integrated unit under test.

To simulate this software conveniently, it is common to use a host-based compiler like Visual Studio, GCC or MinGW to run the code, and once a level of confidence has been achieved the cross-compiler can be used. Depending on the certification level for DO-178C (Level A, B or C), certification credit for the activity may only be permissible when done using the cross-compiler and running on the target.

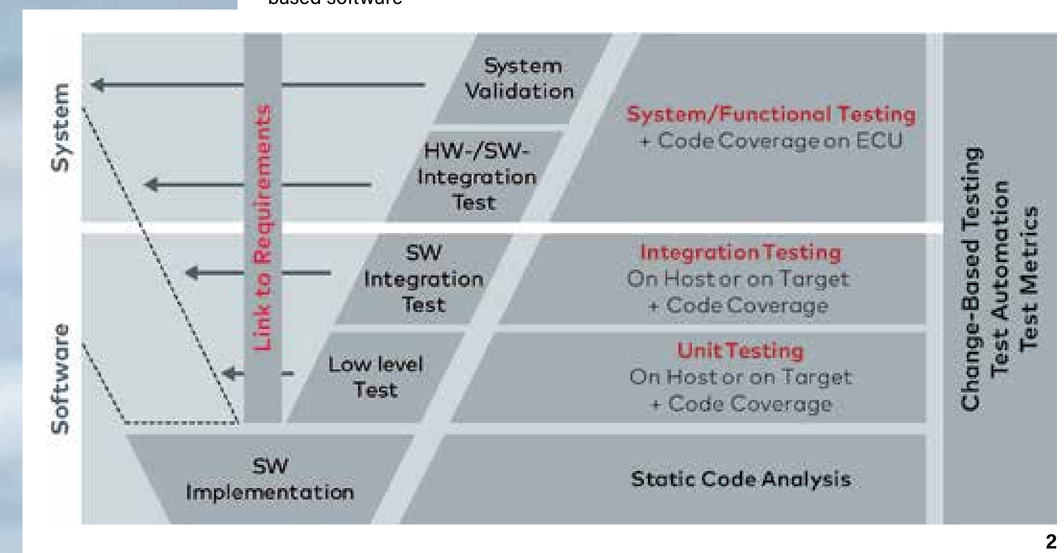
In the low-level testing framework, the software units can still only be tested via programming API calls. In this case a test automation framework like VectorCAST is ideal, as it will automatically build the required drivers and stub any units that are outside the units of interest. There could also be an opportunity to reuse some of the test cases from low-level testing for units that are higher in the call tree.

Alternatively the software components to be tested may be closely reliant on the underlying hardware, and a more robust simulation of the hardware is required to correctly verify the software functionality.

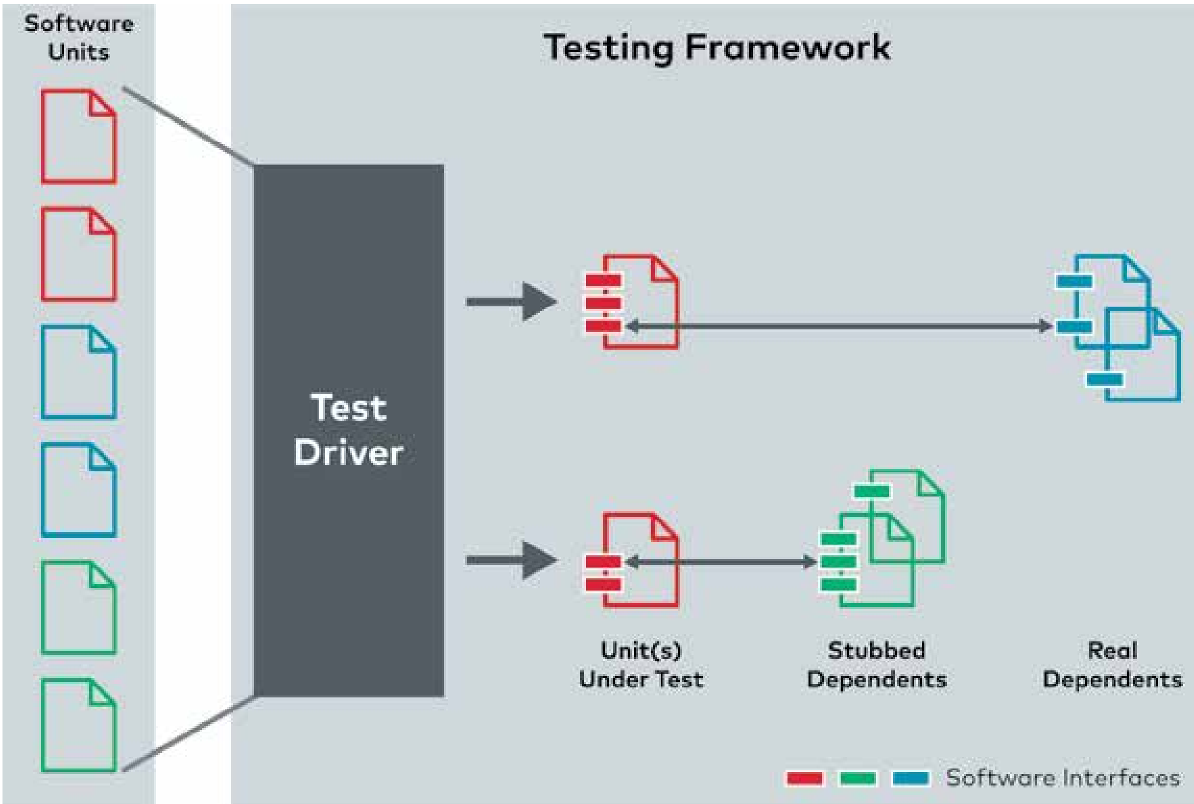
HARDWARE/SOFTWARE INTEGRATION TESTING

This is performed on the target hardware using the complete executable image to satisfy high-level requirements. The challenge when testing at this level is to

2 // The major phases of verification and validation in avionics and ground-based software



2



3

3 // Low-level testing framework to test a single software unit in isolation – using framework code such as test drivers and stubs for dependencies

"A CRITICAL ASPECT OF TESTING SOFTWARE DURING THIS PHASE IS THE ABILITY TO SIMULATE DEPENDENCIES"

provide enough external stimulation to the line replaceable unit (LRU), so that it functions correctly. The external simulation comes in various forms – logical pins, avionics data network and modeling tools. Additionally, because of the complex nature of the networks, it should also be possible to easily extend or customize the simulation interfaces quickly and easily.

An example system to validate an LRU at this level can be set up using the VT System and CANoe tools from Vector (Figure 4). The software and hardware combination offers a test system that can be scaled from simple test equipment at the developer's workstation to a highly automated hardware-in-the-loop (HIL) environment. The core idea of the VT System is to combine all the hardware functions required for LRU testing in a modular system seamlessly integrated into CANoe. The test hardware covers the inputs and outputs, including the power supply and network connections of a control unit or subsystem. At each pin, changing the function according to stimulation, measurement, load simulation, fault connection and switching between simulation and original sensors and actuators is possible. These functions are universally designed to such a degree that, once constructed, a test system can be used for different LRUs.

In CANoe, in addition to the network environment, the physical environment can also be simulated using appropriate MATLAB/Simulink models. A closed HIL simulation is just as possible as a simple, manual stimulation without elaborate models. CANoe offers the same flexibility in test automation, while vTESTstudio provides a modern authoring tool. Tests can be defined in different programming languages like Vector's own CAPL and .NET/C#. Furthermore, test procedures can be described in tabular form or graphically noted test models. It is used to define test procedures and enables the developer to flexibly combine the different input methods. The finished test sequences are stored as test units and then executed in CANoe.

CANoe executes the test cases and at the end of each test run the system creates a detailed test report. Finally, all threads, from test and execution planning to execution documentation, converge in test data management. This ensures the traceability of tested requirements.

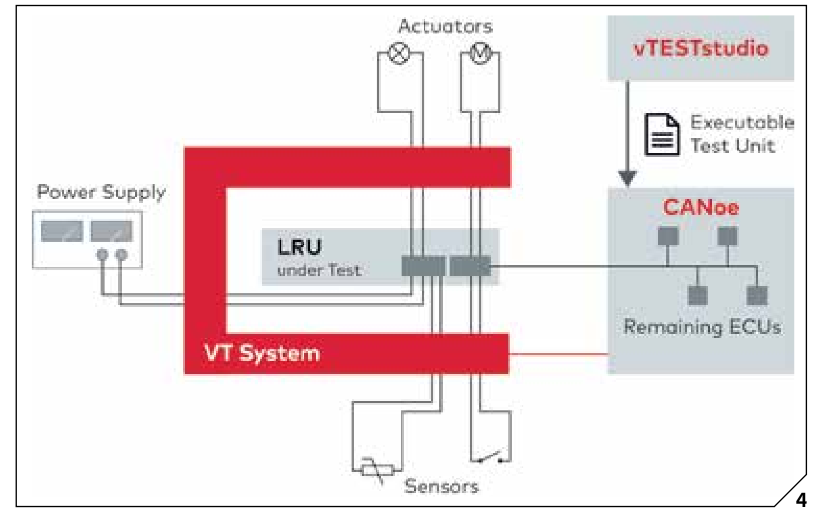
ENABLING A LEAN CONTINUOUS INTEGRATION PIPELINE

One of the biggest challenges with software development is the unintended propagation of defects or issues through the development cycle of a system. These issues can often be identified very early but are sometimes missed because the software is merged without the adequate verification and validation. To address this quality issue, there are various discussions on the topic of shift left. However, in general it can be very time consuming to rerun all low-level software integration and hardware/software integration tests. In some cases a complete end-to-end run of all test cases can take two months. This timeframe does not suit the rapid progress necessary to provide developers with early feedback on issues that they might have introduced when writing the software.

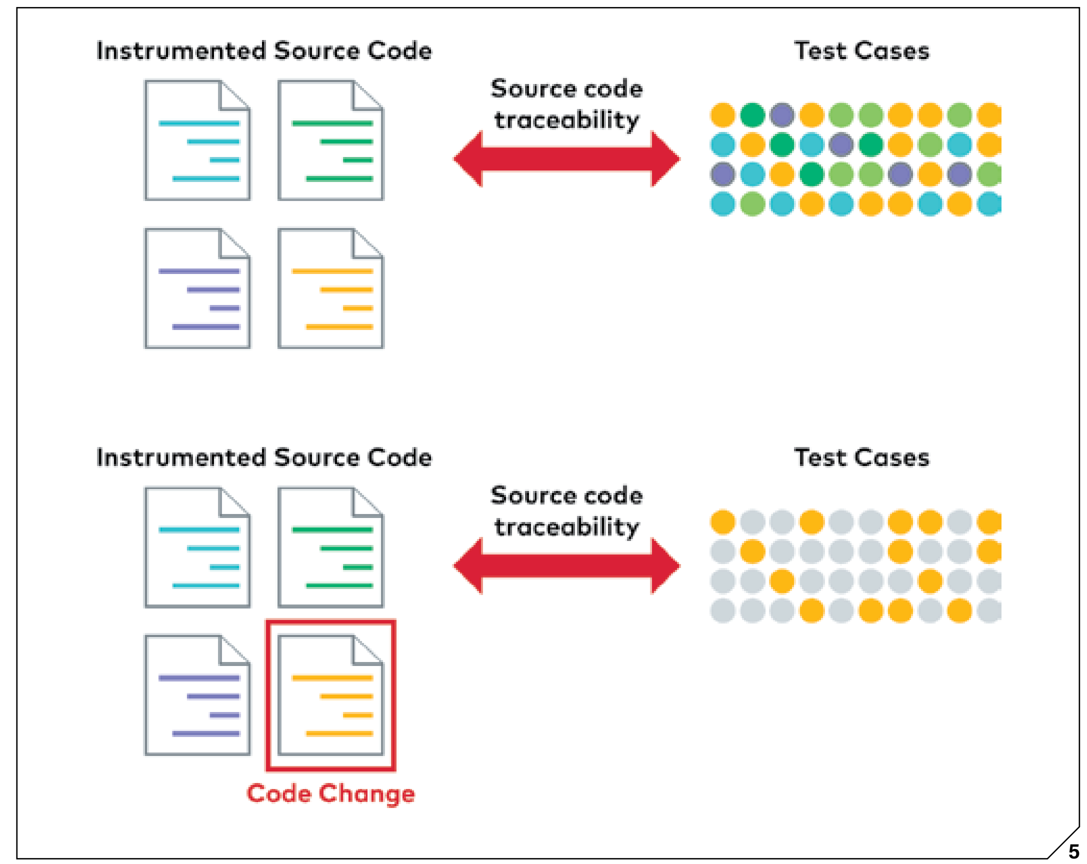
Change-based testing (CBT) can help address this. This method helps organizations test faster and smarter by analyzing each code change against all existing test cases and choosing the subset of tests that are affected by the change (Figure 5). By running only this subset of tests, execution times are greatly reduced and developers get immediate feedback on the impact of their changes. This allows bugs to be fixed as soon as they are introduced, rather than weeks later during full testing.

By using a test automation platform like VectorCAST, structural code coverage is collected from all levels of testing, including low level, software integration and hardware/software integration. The ability to merge code coverage reporting with software integration and hardware/software integration tools such as CANoe and VT System provides a single perspective of the system's aggregated code coverage, and how a specific test contributed to the overall code coverage.

Thus, when a change is made to the underlying software, the VectorCAST decision engine quickly computes the impacted tests at all levels and dispatches them appropriately – even when the test is to be run through CANoe or VT System. Running a subset of tests represents a substantial saving in execution time and shortens to a matter of hours, and with a high level of confidence, the time taken to determine the impact of a change.



4



5

TACKLING TEST COMPLEXITY USING VECTOR TOOLS IN ALL TEST PHASES

As the complexities of avionics and ground-based systems continue to evolve, the need to provide more sophisticated strategies and tooling to address the compliance required for verification and validation for DO-178C and DO-278 will continue to grow. The networked aircraft will require the ability not only to ensure that a single LRU functions correctly, but also that they all function correctly when the entire system is brought together.

This means that the ability to isolate components at a software unit level, as well as an LRU level while simulating the remaining interfaces, will be critical to achieving the quality requirements of the avionics industry.

Furthermore, the artifacts from verification and validation can be introduced into a continuous integration process to introduce modern shift-left concepts into the development of safety critical systems while ensuring compliance with the standards. \\

Dr Arne Brehmer is head of the aerospace business area; Hans Quecke is team leader and product manager for networking and communication design and production; and Niroshan Rajadurai is EMEA director at Vector Informatik

4 // Example setup to simulate the environment around an LRU

5 // Change-based testing greatly reduces testing time while ensuring completeness