LDRA
Software Technology

Technical Briefing

DO-178C: Get on a High with your Software Development

Working with the Airborne Software Industry to Meet the Challenges of Achieving Cost-effective Certification

www.ldra.com

Background

The guidance document DO-178 "Software Considerations in Airborne Systems and Equipment Certification"[1] [2] was first published in 1982, re-written in 1992 as DO-178B and updated in 2011 as DO-178C, to reflect the experience accrued to meet today's aviation industry needs.

LDRA has participated extensively on both the DO-178B[3] and DO-178C[4] committees over nearly two decades. Mike Hennell, LDRA's CEO, was instrumental in the inclusion of several test measurement objectives in the standard, including those relating to structural coverage analysis. The LDRA tool suite® was itself a forerunner in automated verification for certification to both DO-178B, and to its companion standard, DO-278[5] for ground-based systems.

The ARP 4754A[6] standard dictates that functional hazard analyses and system safety assessments are completed prior to a system's development. A Development Assurance Level (DAL) is assigned accordingly for that system, and for the subsystems that implement its hardware and software requirements. The DO-178C standard then provides detailed guidance for the development and verification of safety critical airborne software systems in accordance with the assigned DAL, such that the effort and expense of producing such as a flight control system is necessarily higher than that required to produce (say) a bathroom smoke detector.

DAL	Failure Condition
A	Catastrophic
B	Hazardous
C	Major
D	Minor

DO-178C covers the complete software lifecycle: planning, development and integral processes to ensure correctness and robustness in the software. The integral processes include software verification, software quality assurance, configuration management assurance and certification liaison with the regulatory authorities.

Although the standards do not oblige developers to use analysis, test, and traceability tools in their work, such tools improve efficiency in all but the most trivial projects to the extent that they have a significant part to play in the achievement of the airworthiness objectives for airborne software throughout the development lifecycle. The LDRA tool suite is used to help achieve DO-178C objectives including bi-directional traceability, test management, source code static analysis, and dynamic analysis of both source and object code.

DO-178B Process Objectives

DO-178C recognizes that to ensure correctness, control and confidence in software, functional safety must be addressed systematically throughout the software life cycle.

DO-178C Section 5.0: SOFTWARE DEVELOPMENT PROCESSES

Five high-level processes are identified in the DO-178C SOFTWARE DEVELOPMENT PROCESSES section; Software Requirements Process (5.1), Software Design Process (5.2), Software Coding Process (5.3), Integration Process (5.4), and Software Development Process Traceability (5.5).

Tools used for requirements management (Section 5.1) vary from simple spreadsheets or Microsoft Word documents, to Application Lifecycle Management (ALM) tools such as IBM Rational DOORS[7] and Siemens Polarion PLM[8].

Section 5.3 specifies obligatory software coding process objectives, such as the implementation of low-level requirements and the conformance to a language subset (or "coding standard"). LDRA's static analysis tools make compliance checking easier, less error prone and more cost effective than manual techniques by parsing the code under review with reference to the rules dictated by standard. Non-conformances are highlighted, the complexity of the code under review assessed, and data flow analysis completed to identify any uninitialized or unused variables and/or constants.

[1] http://www.rtca.org/
[2] http://www.eurocae.net/
[3] http://www.rtca.org/store_product.asp?prodid=581
[4] http://www.rtca.org/store_product.asp?prodid=803
[5] http://www.rtca.org/store_product.asp?prodid=678%20%20
[6] http://standards.sae.org/arp475a/
[7] http://www-03.ibm.com/software/products/en/ratidor
[8] https://polarion.plm.automation.siemens.com/

Section 5.5 mandates that artefacts are generated to show that each development phase is complete, coherent, and traceable to its predecessor. The use of an integrated requirements coverage and test management solution addresses the inherent project management challenges. (Figure 1).

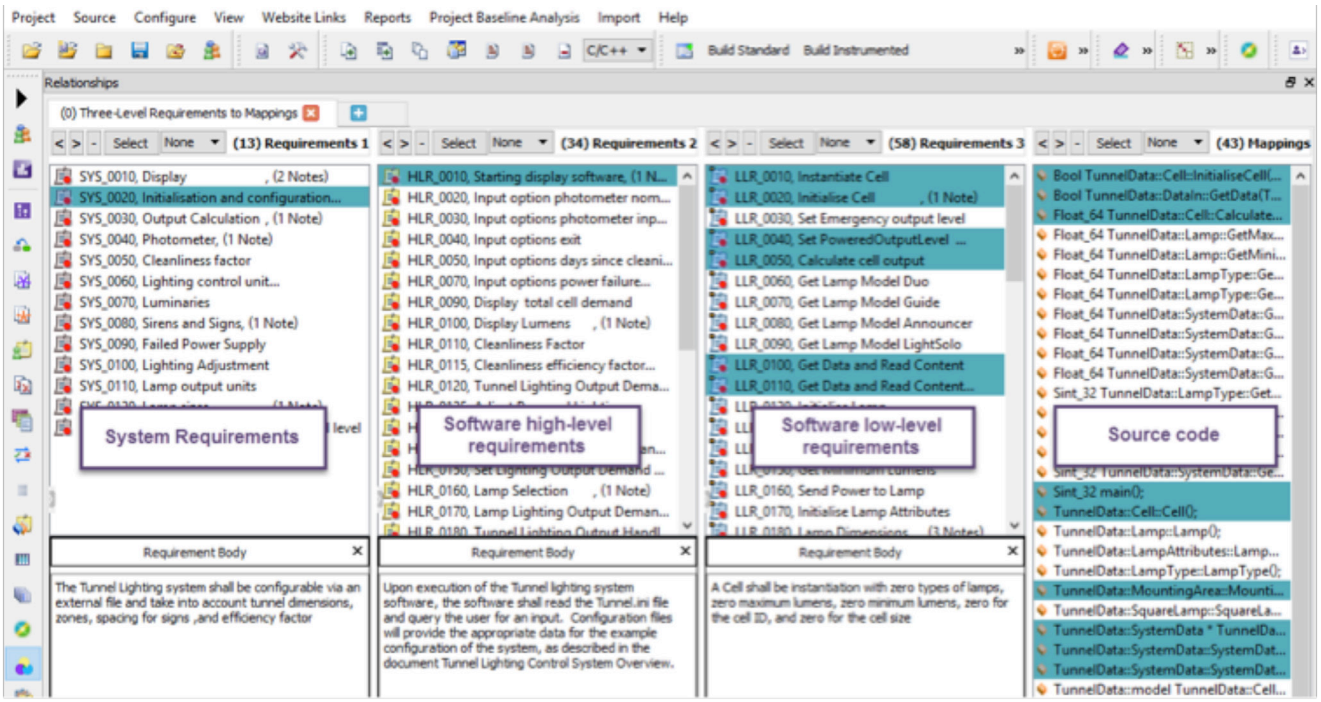

Figure 1: Automating requirements traceability with the TBmanager component of the LDRA tool suite

DO-178C Section 6.0: SOFTWARE VERIFICATION PROCESSES

Dynamic analysis involves using “test cases and procedures” (DO-178C Section 6) to exercise Executable Object Code (EOC), on a target representative of that to be deployed in the field. It provides evidence of both correct functionality and of the parts of the code exercised to achieve it (“structural coverage”). The “test cases and procedures” can include any combination of low-level tests (sometimes referred to as unit tests), integration tests, and system tests.

Low-level tests verify the complete and exclusive implementation of the low-level requirements specified in the Software Verification Plan (SVP), whereas software integration testing verifies the relationships between software components with reference to the requirements and the software architecture. In practice, the mechanisms used for low-level testing often lend themselves to integration testing and hence verify behaviour in the context of a call tree.

Whenever changes are made, all impacted low-level and integration tests need to be re-run (“regression tested”). Regression tests can be re-applied as development progresses to ensure that existing functionality is not compromised by new development. Figure 2 shows an example traceability matrix of high-level requirements versus functional test cases, where one requirement has no associated test case.

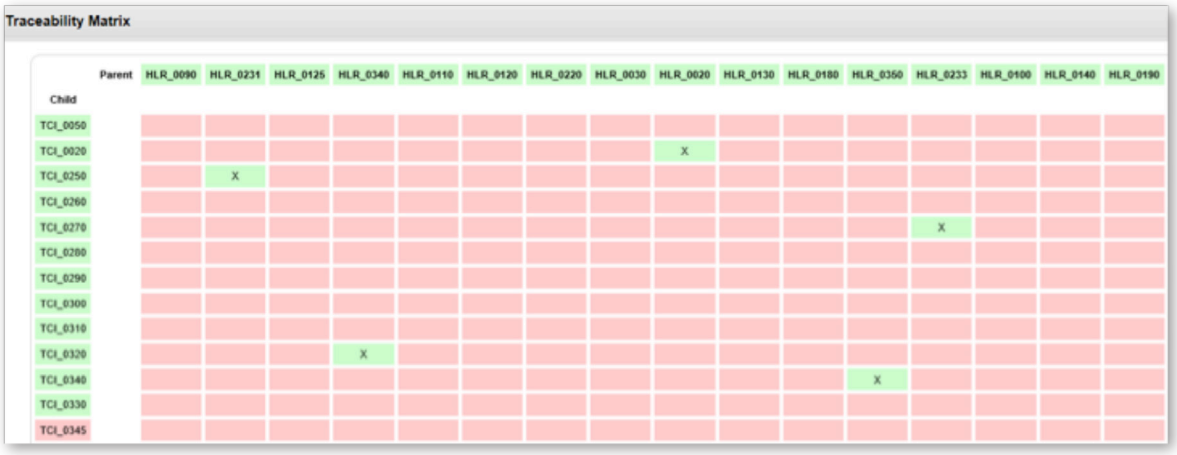

Figure 2: Excerpt from a traceability matrix (High-level requirements to tests cases) as illustrated by TBmanager, a component of the LDRA tool suite

DO-178C Structural Coverage Analysis Objectives

Structural Coverage (SC) data is collated during requirements-based test procedures. A copy of the source code is typically "instrumented" with function calls to show the paths taken during execution, and the resulting output submitted to Structural Coverage Analysis (SCA). Unexecuted sections of the code may require changes to test cases or requirements, or removal of dead code (figure 3). An iterative "review, analyse, verify" cycle is typically needed to ensure that objectives are met.

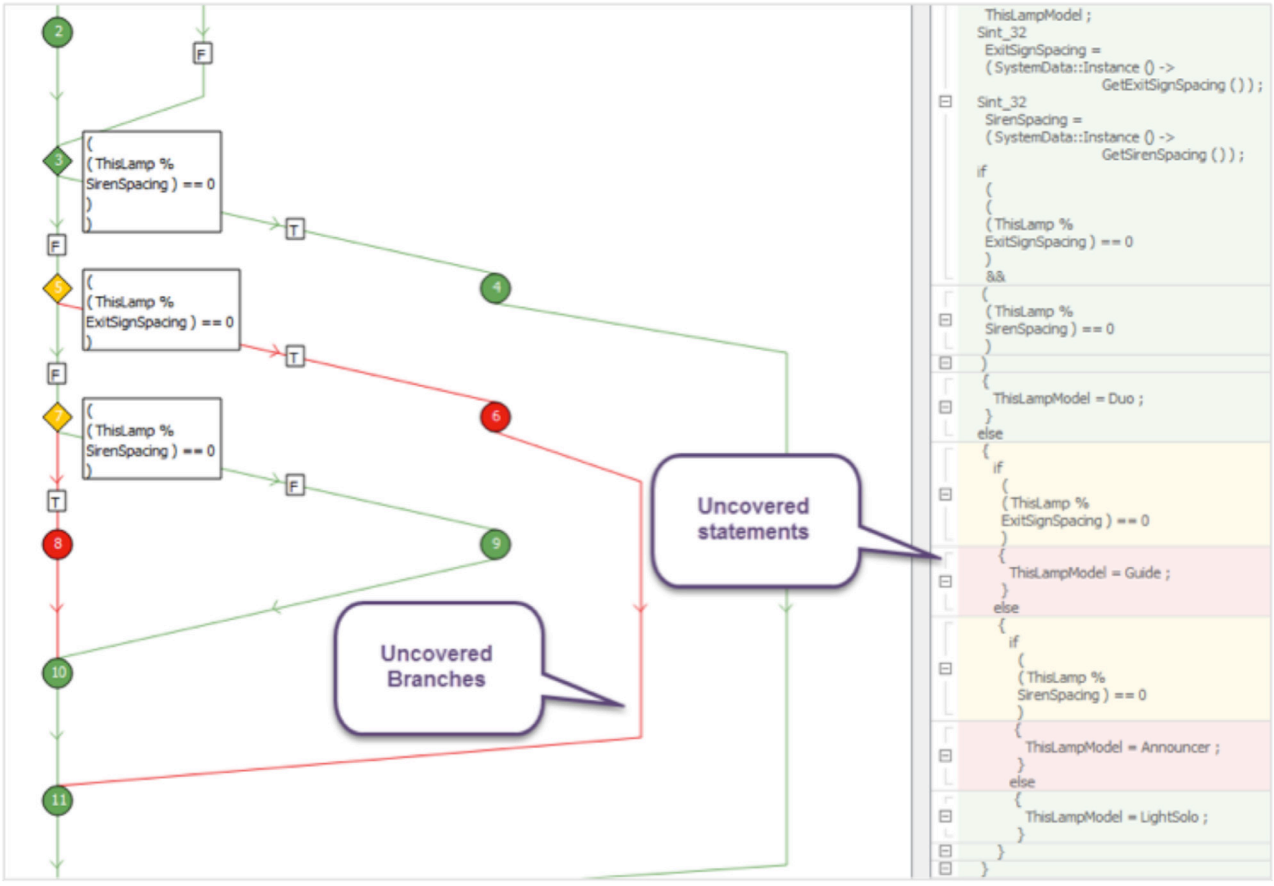

Figure 3: Graphical visualisation of code coverage in a flow graph in the LDRA tool suite

Comprehensive functional test and structural coverage artefacts help support the assertion that system requirements are complete and traceable throughout development.

Structural coverage can take several forms. DO-178C objectives A7-5, 6, and 7 relate to the achievement of 100% MC/DC, decision, and statement coverage respectively. The higher the DAL, the more demanding the structural coverage objectives. For Level A systems, structural coverage at the source level isn't enough. Compilers often add additional code or alter control flow, and often resulting behaviour is not deterministic. DO-178C 6.4.4.2.b states:

"If the software level is A and a compiler, linker, or other means generates additional code that is not directly traceable to Source Code statements, then additional verification should be performed to establish the correctness of such generated code sequences".

An automated approach aids efficiency. The direct one-to-one relationship between object code and assembly code provides a mechanism to achieve Object Code Verification (OCV) (Figure 4). Three discrete modes are used for each test case to quickly identify the "*additional code*".

1. The test case is executed without instrumentation to confirm correct functionality.
2. The test case is executed by leveraging instrumentation at the source code level.
3. Finally, the test case is executed with instrumentation at the assembly code level to identify any uncovered statements or branches that may have been inserted or altered during the compilation and linking process.

Typically, a few additional requirements based tests can be added to verify this additional code to meet objective A7-9 (Figure 4).

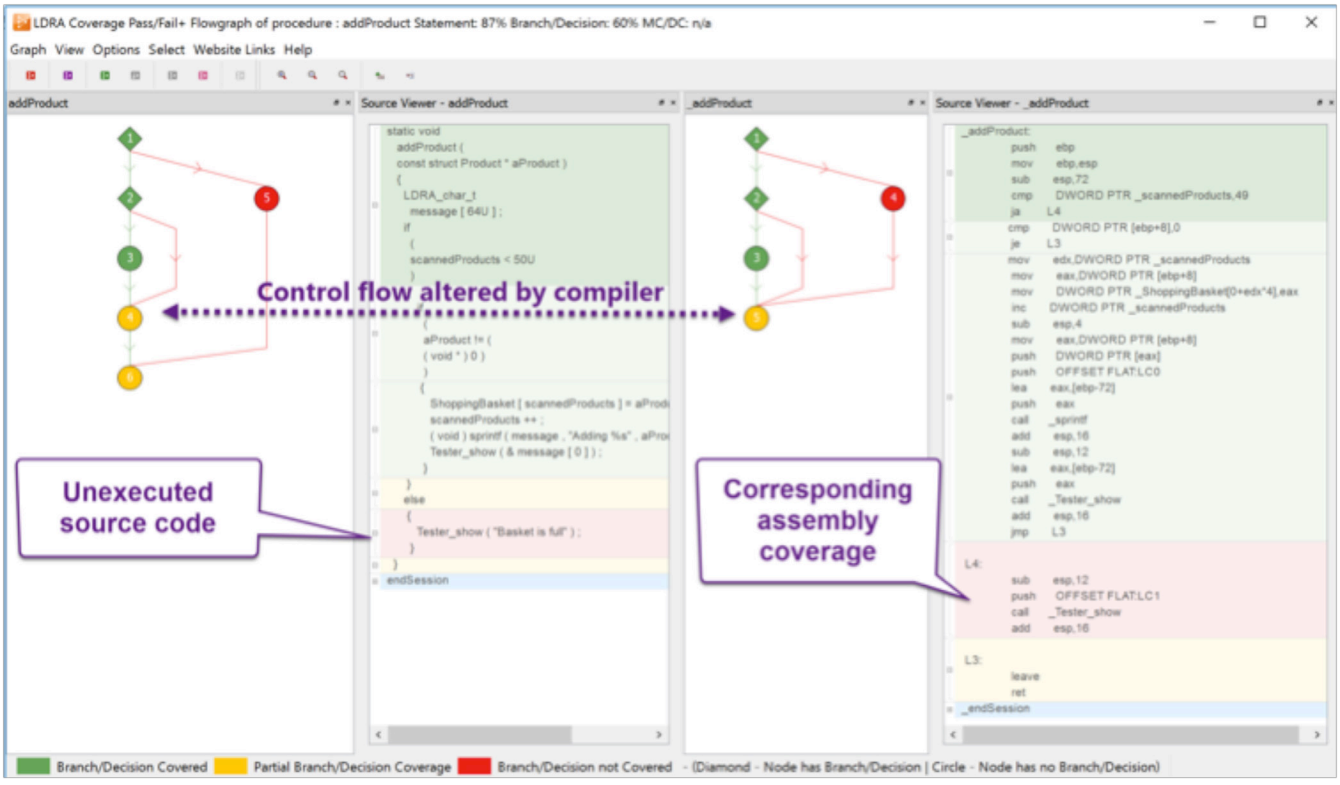

Figure 4: Visualization of control flow and code coverage in C and associated assembly code in the LDRA tool suite

Automated source code instrumentation and coverage data analysis reduces space and time overhead, making the technology scalable and adaptable to a wide array of embedded environments. Target integrations are highly extensible and support processors from simple 8 bit devices to high-performance multi-core architectures, IDEs, and I/O integrations.

Data Coupling and Control Coupling

Control Coupling is defined by DO-178C as "*The manner or degree by which one software component influences the execution of another software component*". Procedure/functional call coverage must be derived from the execution of requirements based tests to identify any gaps and guide targeted verification activities.

SCA and associated artefacts provide visibility and data to perform these activities and meet the associated objective A-7.8.

Data Coupling is defined in the standard as "*The dependence of a software component on data not exclusively under the control of that software component*". Objective A-7.8 requires that "*Test coverage of software structure, both data and control coupling, is achieved*". Any dataflow measurements must be derived from requirements based tests. The example in figure 5 is duplicated from DO-248C, and its implementation is illustrated in the function *runAirspeedCommand* on the right. The code is expected to first calculate the airspeed and then display it.

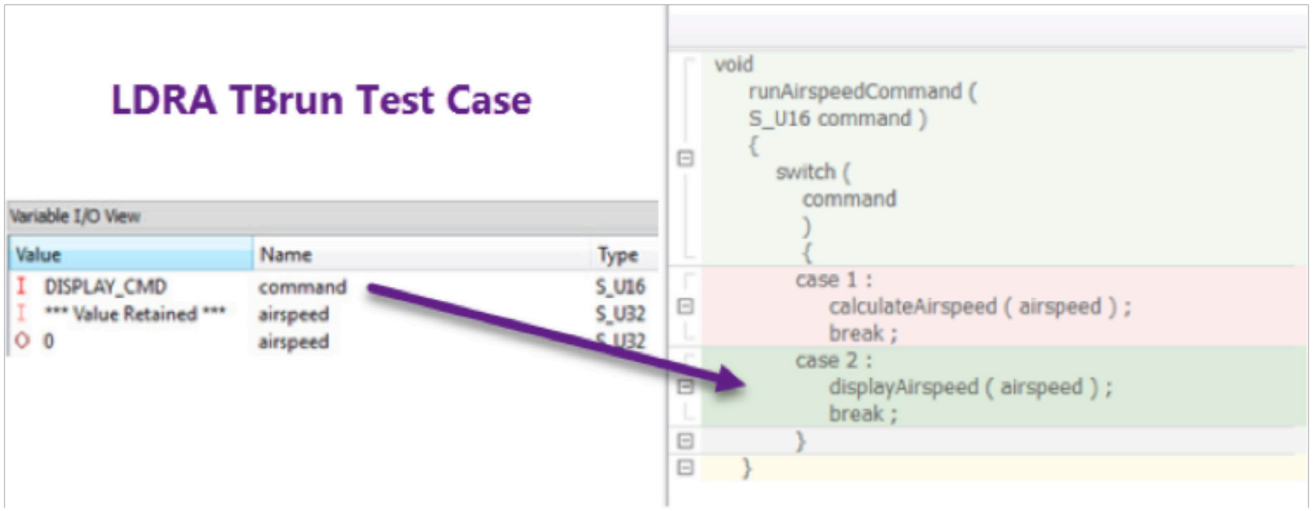

Figure 5: Test case exercising runAirspeedCommand with the resulting control flow and structural coverage represented in green. Example from TBrun, a component of the LDRA tool suite

The test exercises the second case in the switch statement as reflected by the structural coverage. It also shows that display is called without updating the airspeed to its latest value. Additional test cases may invoke the *calculateAirspeed* command but not necessarily after a call to *displayAirspeed*.

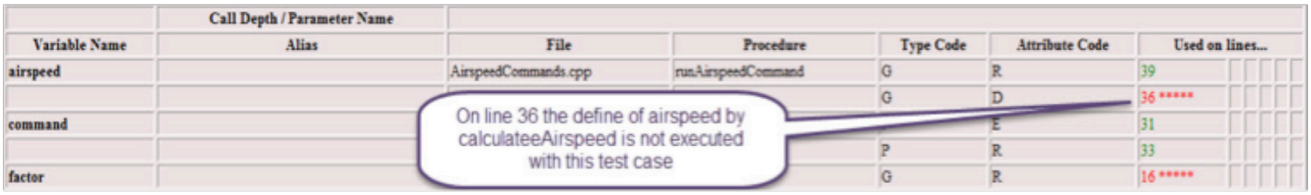

Figure 6: LDRA tool suite report showing unreferenced variable in run time. These artefacts are used to meet objective A-7.8

The report in Figure 6 shows dynamic data flow information resulting from that test case. It reveals that airspeed was in fact not written to on line 36 and wasn't updated before it was displayed. Data coupling analysis is focused on the observation and analysis of data elements such as *airspeed*, as they are set and used ("set/use pairs") across software component boundaries.

Object-Oriented Technology

Supplement DO-332[9] to DO-178C describes key features of object-oriented technologies and related techniques, discusses their impact on the planning, development, and verification processes, and enumerates their vulnerabilities. Two objectives were included in the DO-332 supplement.

A-7 OO.10 Verify local type consistency (OO.6.7.1)

Ensuring that that "*each class passes all the tests of all its parent types which the class can replace*" is usually the most practical approach here. Consider a superclass *Rectangle* whose methods explicitly set the height and width, and a *Square* subclass that sets both in the *SetWidth* function. Type consistency is violated because test cases for the *SetWidth* method of the *Rectangle* class may not pass those for its subtype *Square*.

In this example, type inconsistences can be highlighted by applying the same test cases to the parent class *Rectangle* and the subclass *Square*.

A-7 OO.11 Verify the use of dynamic memory management is robust (OO.6.8.1)

A range of static and dynamic analysis techniques can be deployed in order to fulfil DO-332 A-7 OO.6.8.1 and the related vulnerabilities outlined in Annex OO.D.1.6.1.

Tracking memory allocation and deallocation helps to ensure the proper freeing of memory, as do associated checks prior to dereferencing. Low-level testing provides a mechanism to explore various allocation/deallocation scenarios to help ensure that vulnerabilities are addressed. Timing hooks within low-level tests help characterize allocation/deallocation timing, and dynamic data flow analysis monitors data elements in runtime to detect lost updates and stale references.

Model-Based Development

Supplement DO-331[10] to DO-178C addresses Model-Based Development (MBD), asserting that specification models or design models take the place of high-level and low-level requirements respectively. Textual requirements may be linked to models upstream or downstream.

Popular tools such as MathWorks® Simulink®[11], IBM® Rational® Rhapsody®[12], and ANSYS® SCADE[13] can generate code automatically. DO-331 MB.5.0 (Software Development Processes) addresses traceability, model standards and more for both software requirements and design processes where such tools are used. MB.5.3 (Software Coding Processes) is simply a cross-reference to DO-178C, emphasizing that best-practice coding-related process activities still apply.

Projects using auto-generated code almost always contain some hand-code too. It is possible to apply different coding standards to these different code subsets, such as MISRA C:2012 for hand-code, MISRA C:2012 Appendix E for auto-generated code, and a custom coding standard for legacy code.

DO-331 MB 6.0 (Software Verification Process) expands on how best practice applies to MBD, with DO-331 MB.6.8.2 (Model Simulation for Verification of Executable Object Code) expanding upon which verification objectives must be performed at the target level.

The objectives listed requiring at least some verification at the target level include EOC robustness, compliance with low-level requirements, test coverage of low-level requirements, confirmation of compatibility with the target hardware, and hardware/software integration testing. The supplement also lists various types of errors that can only be detected on the target hardware.

[9] RTCA DO-332 Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A

[10] RTCA DO-331 Model-Based Development and Verification Supplement to DO-178C and DO-278A

[11] https://uk.mathworks.com/products/simulink.html

[12] http://www-03.ibm.com/software/products/en/rstirhapfami

[13] http://www.ansys.com/products/embedded-software/ansys-scade-suite

DO-331 MB.B.11 (FAQ #11) addresses the questions of model coverage activity, stating that model coverage analysis can be considered in very specific scenarios, "*...on a case-by-case basis and agreed upon by the certification authorities...*". As a result, most organizations do some of the verification activities within the model but then re-affirm the results of those activities on the target.

The integration of the LDRA tool suite and modelling tools help to achieve that seamlessly, including the static analysis of generated code, the collection of code coverage from model execution, and the migration of model tests into an appropriate form for execution on the target hardware.

Tool Qualification

If software tools are to automate significant numbers of DO-178C activities while producing evidential artefacts showing that objectives have been met, it is essential to ensure that those tools can be relied upon. DO-178C states that:

"The purpose of the tool qualification process is to ensure that the tool provides confidence at least equivalent to that of the processes of this document are eliminated, reduced, or automated".

Tool qualification is a vital part of to the certification process, and it is documented in the supplement Software Tool Qualification Considerations (DO-330)[14]. Where a tool is designed to be used for verification purposes, it is assigned Tool Qualification Level 5 because its output is not used as part of the airborne software and therefore cannot introduce errors.

Certification authorities such as the FAA, CAA, JAA, and ENAC undertake tool qualification on a project by project basis, so the responsibility for showing the suitability of any tools falls on to the organisation developing the application. However, they can make use of Tool Qualification Support Packages (TQSP) provided by the vendor. Tool Qualification documentation must be referenced in other planning documents, and it plays a key role in the compliance process.

Tool Selection

The use of traceability, test management and static/dynamic analysis tools for an airborne software project that meet the DO-178C certification requirements offers significant productivity and cost benefits. Tools make compliance checking easier, less error prone and more cost effective. In addition, they make the creation, management, maintenance and documentation of requirements traceability straightforward and efficient. When selecting a tool to assist in achieving DO-178C acceptance the following criteria should be considered:

- Does the tool provide a complete 'end-to-end' traceability across the lifecycle through requirements, code, tests, artefacts, and objectives?
- Does the tool provide static analysis to ensure conformance to industry leading coding standards?
- Does the tool enable Structural Coverage Analysis on the target hardware, as laid out in section 6.4.4.2 of the standard, including coverage at the source and object levels for Level A projects?
- Is the tool available for all applicable languages, platforms, tool chains, and targets?
- Has the tool been utilized in this manner successfully already?
- Will the tool vendor assist in tool qualification?
- Is tool support both flexible and extensive enough to meet changing requirements?
- Is the tool easy to use?

[14] RTCA DO-330 Software Tool Qualification Considerations Supplement to DO-178C and DO-278A

www.ldra.com

LDRA UK & Worldwide
Portside, Monks Ferry,
Wirral, CH41 5LH
Tel: +44 (0)151 649 9300
e-mail: info@ldra.com

LDRA Technology Inc.
2540 King Arthur Blvd, 3rd Floor, 12th Main Lewisville Texas 75056
Tel: +1 (855) 855 5372
e-mail: info@ldra.com

LDRA Technology Pvt. Ltd.
Unit B-3, Third floor Tower B, Golden Enclave
HAL Airport Road Bengaluru 560017
Tel: +91 80 4080 8707
e-mail: india@ldra.com