LDRA
Software Technology

Technical Briefing

SAE J3061 and ISO 26262?
They're Made for Each Other.

Cost Effective Software Certification from ASIL A to ASIL D

www.ldra.com

* Registration required to download the document

Background

Over the past few years, there has been a proliferation of automotive electrical and/or electronic (E/E/PE) systems such as adaptive driver assistance systems, anti-lock braking systems, steering and airbags. ISO 26262 "Road vehicles – Functional safety" was first published in 2012 in response this explosion in automotive E/E/PE system complexity and the associated risks to public safety[1], bringing with it an opportunity for automotive manufacturers to embrace best functional safety practices throughout the development lifecycle.

ISO 26262 requires any threats to functional safety to be adequately addressed, implicitly including those relating to security threats, but it gives no explicit guidance relating to cybersecurity.

At the time of ISO 26262's publication, that was perhaps to be expected. Automotive embedded applications have traditionally been isolated, static, fixed-function, device-specific implementations, and practices and processes have relied on that status. But the rate of change in the industry has been such that by the time of its publication in 2016, SAE International's Surface Vehicle Recommended Practice SAE J3061™ in January 2016 was much anticipated[2].

SAE J3061 and ISO 26262

SAE J3061 can be considered complementary to ISO 26262 in that it provides guidance on best development practices from a cybersecurity perspective, just as ISO 26262 provides guidance on practices to address functional safety.

SAE J3061 calls for a similar sound development process to that of ISO 26262. For example, hazard analyses are performed to assess risks associated with safety, whereas threat analyses identify risks associated with security. The considerations for the resulting security requirements for a system can be incorporated in the process described by ISO 26262, part 8, section 6: "Specification and management of safety requirements".[3]

Another parallel can be found in the use of static analysis, which is used in safety critical system development to identify construct, errors, and faults that could directly affect primary functionality. In cybersecurity critical system development, static code analysis is used instead to identify potential vulnerabilities in the code.

Beyond Functional Safety

Despite the clear synergy between the two standards, it is important to note that SAE J3061 does more than simply formalize the need to include security considerations in functional safety requirements. Much of this white paper focuses on an appropriate process once functional, safety and security requirements are established but the significance of malicious intent in the definition of those requirements should not be underestimated.

SAE J3061 emphasizes this point throughout. It argues that cybersecurity is likely to be even more challenging than functional safety, stating that *"Since potential threats involve intentional, malicious, and planned actions, they are more difficult to address than potential hazards. Addressing potential threats fully, requires the analysts to think like the attackers, but it can be difficult to anticipate the exact moves an attacker may make."*

Perhaps less obviously, the introduction of cybersecurity into an ISO 26262-like formal development implies the use of similarly rigorous techniques into applications that are NOT safety critical – and perhaps

[1] https://www.iso.org/news/2012/01/Ref1499.html
[2] SAE International Surface Vehicle Recommended Practice – J3061TM - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems. Issued 2016-01.
[3] ISO 26262-8:2011(E) Road Vehicles – Functional safety – Part 8: Supporting processes

into organizations with no previous obligation to apply them. SAE J3061 discusses privacy in general and Personally Identifiable Information (PII) in particular, and highlights both as key targets for a bad actor of no less significance than the potential compromise of safety systems. In practical terms, it therefore demands that ISO 26262-like rigour is required in the defence of a whole manner of personal details potentially accessed via a connect car, including personal contact details, credit card and other financial information, and browse histories. It could be argued that this is an extreme example of the general case cited by the SAE J3061 standard, which states that *"...there is no direct correspondence between an ASIL rating and the potential risk associated with a safety-related threat."*

Not only does SAE J3061 bring formal development to less safety-critical domains, it also extends the scope of that development far beyond the traditional project development lifecycle. Consideration for an incident response process, over-the-air (OTA) updates, and changes of vehicle changes ownership are all examples of that.

An Overview of the Cybersecurity Processes

The cybersecurity lifecycle begins with the development of the Cybersecurity Program Plan to describe the activities to be carried out. Threat analysis and risk assessments (TARA) are used to identify and assess the potential threats to the system and to determine the risk associated with each identified threat. The TARA results then drive subsequent activities by focusing analyses on the highest risk cybersecurity threats.

The product development phase of the lifecycle is subdivided into system, hardware and software levels as reflected in Figure 1 which outlines the principles of an ISO 26262 compliant cybersecurity process. It shows the relationship between the concept and product development phases, and the system, hardware and software product development phases.

Applying SAE J3061 Cybersecurity Processes in Accordance with ISO 26262

The standards lend themselves to integration at each stage of the product lifecycle – even to the extent that the same test team could be deployed to fulfil both safety and cybersecurity roles.

For example, it is possible to perform concurrent hazard analysis, safety risk assessment, threat analysis, and security risk assessment, using a single integrated template and method.

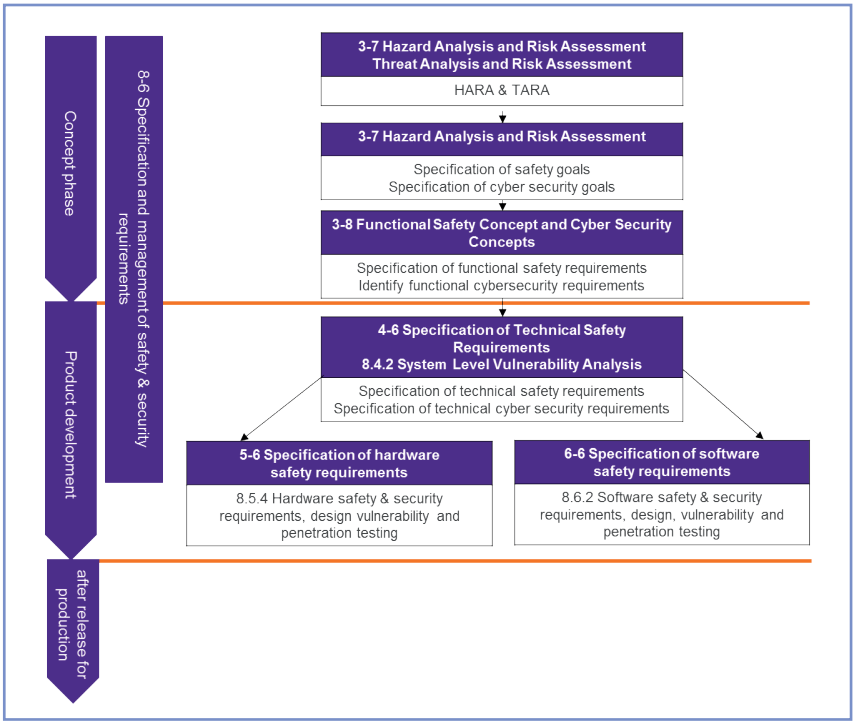

Figure 1: An ISO 26262 compliant cybersecurity process

Software tools vendors such as LDRA have long experience in easing the path to certification both in the automotive sector and elsewhere, and that expertise can be leveraged in the development of SAE J3061 and ISO 26262 compliant applications (Figure 2). Some key elements of that process are discussed in the following.

SAE J3061 Section 8.6.2: "*Specification of Software Cybersecurity Requirements*"

The primary objective here is to review and update the cybersecurity requirements previously identified in the context of the system as a whole. This activity will reference such as hardware and software interfaces, data flow, data storage, data processing, and any subsystems supporting cybersecurity functionality.

SAE J3061 Section 8.6.3: "*Software Architectural Design*"

The objective of this phase is to develop a software architectural design that realizes the security requirements, leveraging the analysis of the "*data types being used, how the data will flow, how the software will detect errors; and how the software will recover from errors*".

LDRA's static analysis tools have a part to play in the verification of the design by reference to the control and data flow analysis of the resulting code. The tools derive the relationship between some or all the code components and represent it graphically such that it can then be compared with the design.

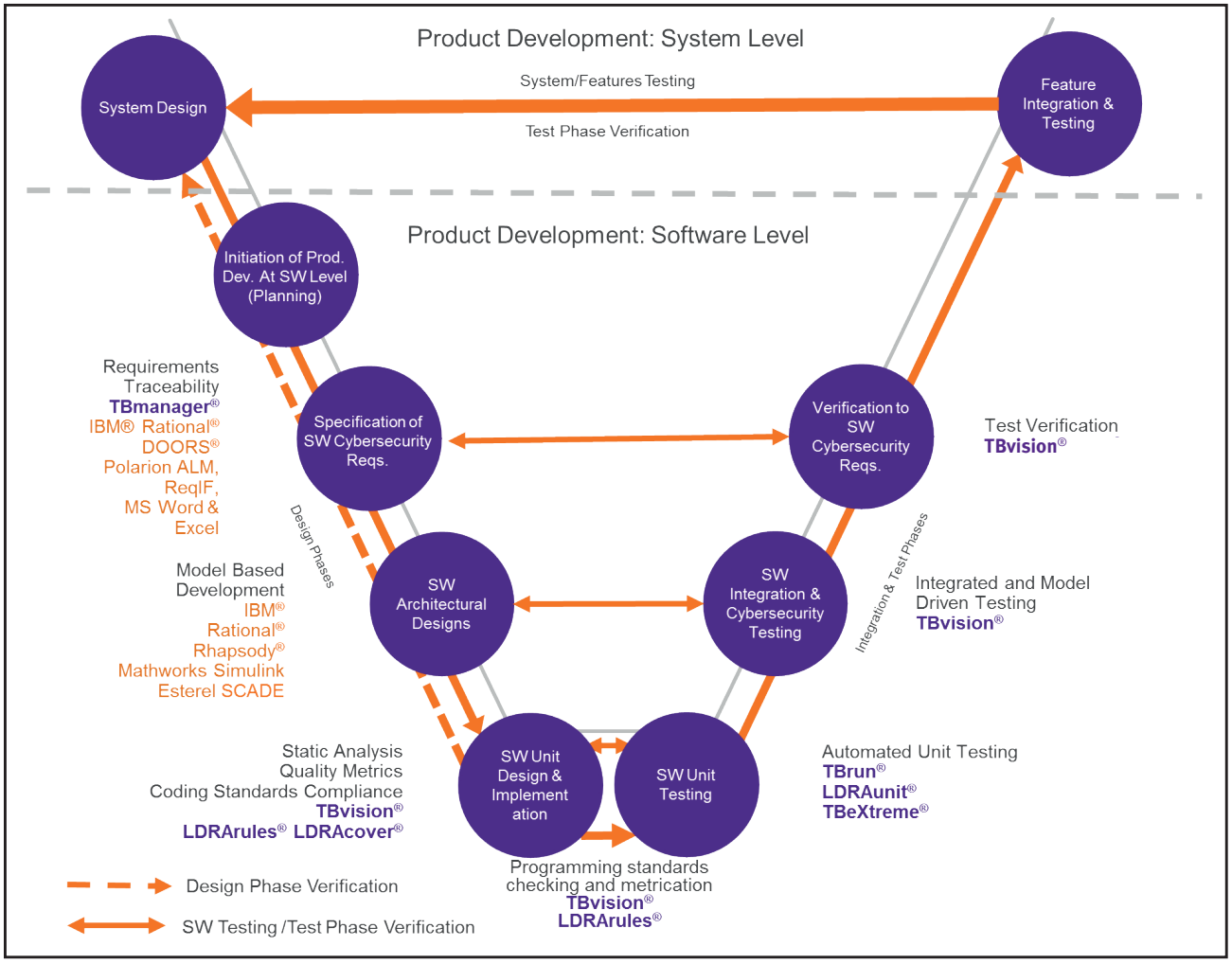

Figure 2: Mapping the capabilities of an automated tool chain to SAE J3061 process guidelines

SAE J3061 Section 8.6.4: "*Software Vulnerability Analysis*"

This is a security specific activity, and so it is supplementary to the processes specified by ISO 26262: 2011. SAE J3061 refers to "Trust Boundaries", which can be thought of lines drawn through a program. On one side of the line, data is untrusted. On the other side of the line, data is assumed to be trustworthy. The objective of Software Vulnerability Analysis (SVA) is to perform an analysis of the software cybersecurity requirements and the data flow from the software architectural design, to define where these trust boundaries exist.

The first step in a software vulnerability analysis is to decompose the application, and to analyse the data and control entry and exit points. The second step involves using threat categorization techniques to identify threats based on that break-down of the system. In some models, severity values for process, control, and data areas are determined while in others, any entry point or trust boundary is treated as critical for the final step.

SAE J3061 Section 8.6.5: "*Software Unit Design and Implementation*"

Coding Guidelines

Again complementing ISO 26262 (Figure 3), secure coding best practices include the application of secure coding standards such as MISRA, CERT or CWE. It is entirely possible for language subsets to encompass both functional safety and cybersecurity. MISRA C:2012 in combination with Amendment 1[4] is designed to do just that. Although many subsets such as CERT C are more focused on one or the other, LDRA Static Analysis test tools can be configured to combined rules from more than one source.

Establishing appropriate project guidelines for coding, architectural design and unit implementation are three discrete tasks but software developers responsible for implementing the design need to be mindful of them all concurrently.

Topics		ASIL			
		A	B	C	D
1a	Enforcement of low complexity	++✓	++✓	++✓	++✓
1b	Use of language subsets	++✓	++✓	++✓	++✓
1c	Enforcement of strong typing	++✓	++✓	++✓	++✓
1d	Use of defensive implementation techniques	o	+✓	++ ✓	++✓
1e	Use of established design principles	+✓	+✓	+✓	++✓
1f	Use of unambiguous graphical representation	+	++	++	++
1g	Use of style guides	+✓	++✓	++✓	++✓
1h	Use of naming conventions	++✓	++✓	++✓	++✓

"++" The method is highly recommended for this ASIL.
"+" The method is recommended for this ASIL.
"o" The method has no recommendation for or against its usage for this ASIL.
✓ Satisfied by the LDRA tool suite

Figure 3: Mapping the capabilities of the LDRA tool suite to "Table 1: Topics to be covered by modelling and coding guidelines" specified by ISO 26262-6:2011[5]

[4] MISRA C:2012 Amendment 1 – Additional security guidelines for MISRA C:2012. April 2016
[5] Based on table 1 from ISO 26262-6:2011, Copyright© 2015 IEC, Geneva, Switzerland. All rights acknowledged.

Software Architectural Design and Unit Implementation

If the coding guidelines of ISO 26262-6:2011 are analogous to the "bricks" of the software application, then its principles for software architectural design define where the "walls" should be built, and its unit implementation guidelines define how those "walls" should be built.

The architectural design and unit implementation principles required by ISO 26262-6:2011 can usually be checked by means of static analysis, just like coding guidelines.

Topics		ASIL			
		A	B	C	D
1a	One entry and one exit point in subprograms and functions	++✓	++✓	++✓	++✓
1b	No dynamic objects or variables, or else online test during their creation	+✓	++✓	++✓	++✓
1c	Initialization of variables	++✓	++✓	++✓	++✓
1d	No multiple use of variable names	+✓	+✓	++✓	++✓
1e	Avoid global variables or else justify their usage	+✓	+✓	++✓	++✓
1f	Limited use of pointers	o	+✓	+✓	++✓
1g	No implicit type conversions	+✓	++✓	++✓	++✓
1h	No hidden data flow or control flow	+✓	++✓	++✓	++✓
1i	No unconditional jumps	++✓	++✓	++✓	++✓
1j	No recursions	+✓	+✓	++✓	++✓

"++" The method is highly recommended for this ASIL.
"+" The method is recommended for this ASIL.
"o" The method has no recommendation for or against its usage for this ASIL.
✓ Satisfied by the LDRA tool suite

Figure 4: Mapping the capabilities of the LDRA tool suite to "Table 8: Design principles for software unit design and implementation" specified by ISO 26262-6:2011[6]

SAE J3061 Section 8.6.6: "*Software Implementation Code Reviews*" and SAE J3061 Section 8.6.10: "*Software Vulnerability Testing*"

Although the flow of the SAE J3061 document sees software implementation code reviews following logically from software integration, they are usually applied in tandem such that the code is reviewed as it is written. A fully integrated tool suite automates that process, helping to ensure that the good practices required by ISO 26262:2011 and SAE J3061 are adhered to whether they are coding rules, design principles, or principles for software architectural design.

Metrics relating to such as software component size, complexity, cohesion, and coupling are more concerned with the structure of the code base rather than the use of particular constructs, and so for them to be meaningful the code needs to be a little more complete. Complexity metrics are generated as a product of interface analysis, cohesion evaluated through data object analysis, and coupling through data control coupling analysis (Figure 5).

SAE J3061 suggests that "Code reviews should be conducted on the software throughout the software design and implementation phase." Traditional peer code reviews may well still have a place in the development process – they can be very useful as an aid to learning between team members, for example – but automating the more tedious checks is far more efficient and less prone to error.

[6] Based on table 1 from ISO 26262-6:2011, Copyright© 2015 IEC, Geneva, Switzerland. All rights acknowledged.

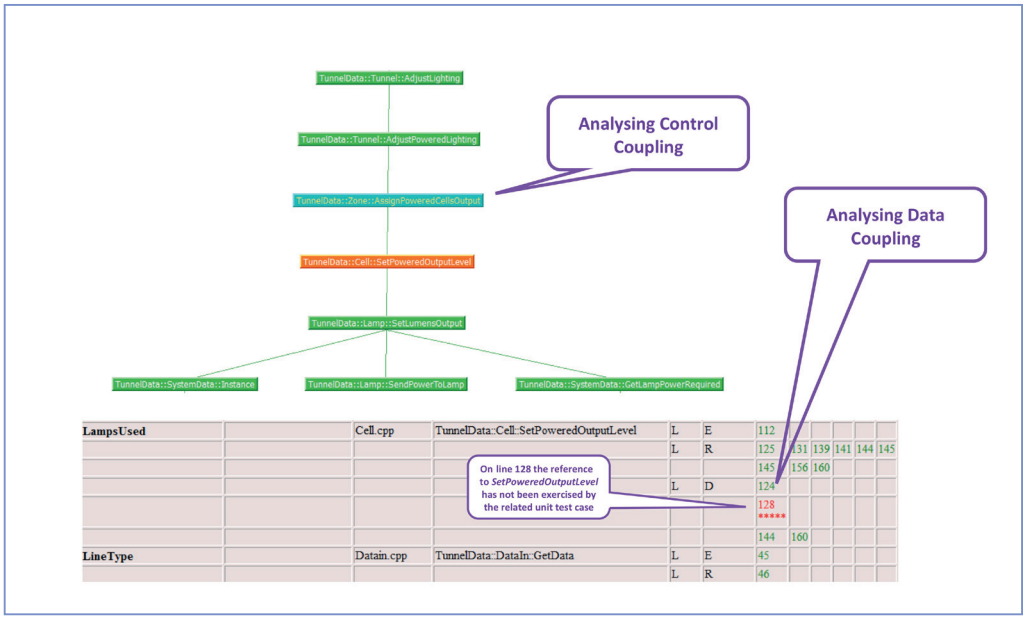

Figure 5: Output from control and data coupling analysis as represented in the LDRA tool suite

SAE J3061 Section 8.6.7: "*Software Unit Testing*" *and* SAE J3061 Section 8.6.8: "*Software Integration and Testing*"

Dynamic analysis techniques (which involve the execution of some or all of the code) are helpful in both software unit testing, and system integration and testing.

Test procedures need to be authored, reviewed, and executed to ensure the software unit meets design criteria but does not contain any undesired functionality. Unit tests can then be executed on the target hardware or simulated environment based on the verification plan and verification specification. Once the test procedures are executed actual outputs are captured and compared with the expected results. Pass/Fail results are then reported and software safety requirements are verified accordingly.

Integration testing is designed to ensure that when the units are working together in accordance with the software architectural design, they meet the related specified requirements. In practice, these integration tests typically involve the verification of functional, safety and cybersecurity functions.

It is desirable for all dynamic testing to use environments which correspond closely to the target environment and hence test dependencies between hardware and software. However, that is not always practical. One approach involves developing the tests in a simulated environment and then, once proven, re-running them on the target.

Neither standard insists that any of these tests deploy software test tools. However, once again, such tools are capable of making the test process far more efficient.

Figure 6 shows how the software interface is exposed at the function scope allowing the user to enter inputs and expected outputs as the basis for a test harness, which is compiled and executed on the target hardware. Actual outputs are captured and compared.

SAE J3061 recommends performing penetration (or "pen") testing and fuzz testing in accordance with software security requirements. Pen testing is usually a system test techniques, performed late in the lifecycle.

Robustness testing is closely related to fuzz testing, and is complementary to it in that it can be performed at unit or integration level, and can generate structural coverage metrics. Both robustness and fuzz testing are designed to test response to unexpected inputs. Robustness testing can incorporate techniques such as boundary value analysis, conditional value analysis, error guessing, and error seeding techniques.

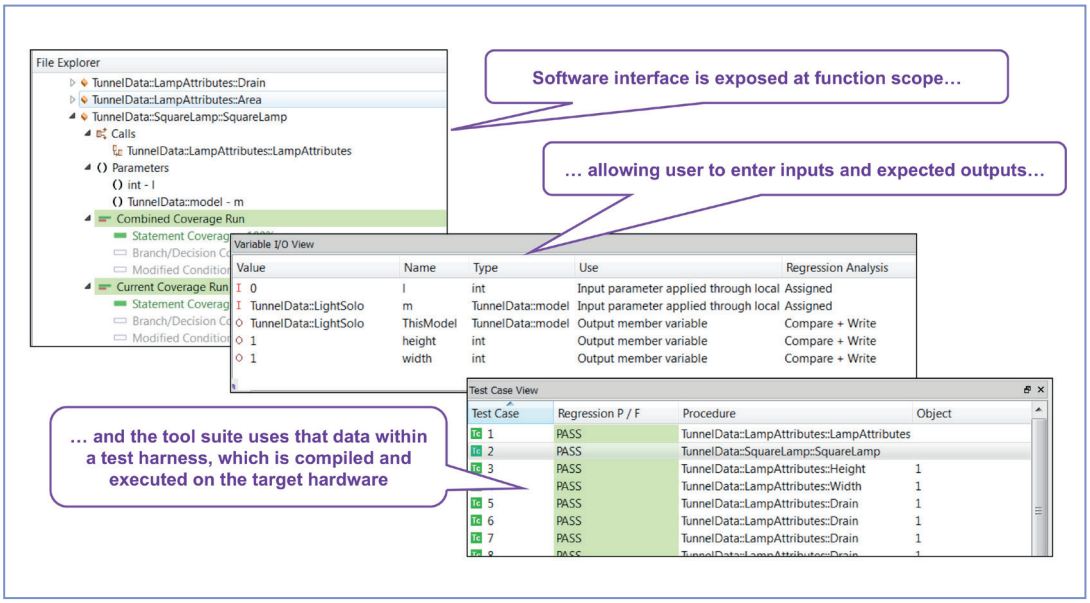

Figure 6: Performing requirement based unit-testing using the LDRA tool suite

Structural Coverage Metrics

System and unit test can also operate in tandem, so that (for instance) coverage can be generated for most of the source code through a dynamic system test and complemented using unit tests to exercise normally inaccessible code, such as defensive constructs (Figure 7).

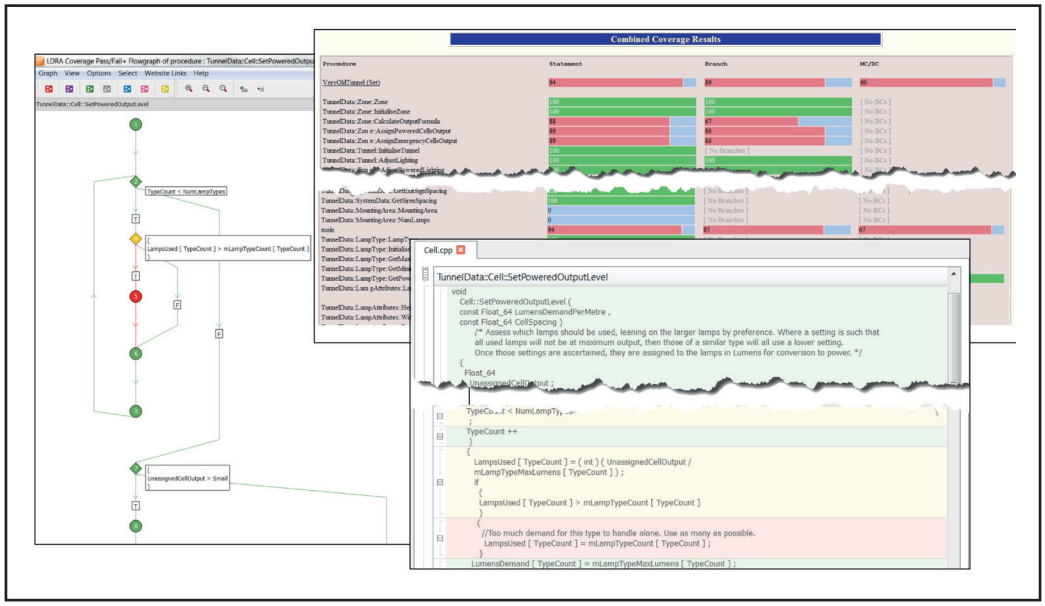

Figure 7: Examples of representations of structural coverage within the LDRA tool suite

SAE J3061 Section 8.6.9: "*Verification/Validation to Software Cybersecurity Requirements*"

Bi-directional traceability is referenced throughout ISO 26262:2011, and SAE J3061 states that the software cybersecurity design is to be "*... traceable to, and validated, against the software cybersecurity requirements*" and that the implementation is to be"*...traceable to and validated against the software cybersecurity design*". Tools can help in establishing the traceability from system level requirements, through high-level requirements and low level requirements, and on to source code and test cases
(Figure 8).

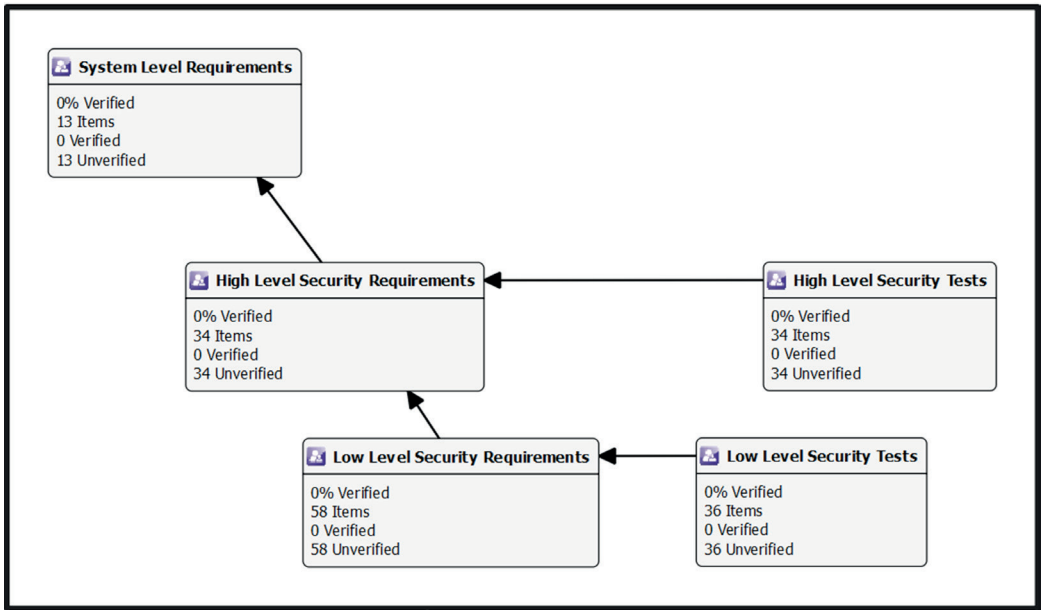

Figure 8: The Uniview graphic from the TBmanager component of the LDRA tool suite, showing the relationships between cybersecurity tests and requirements

Confidence in the Use of Software Tools

ISO 26262-8:2011 section 11 defines a mechanism to provide evidence that the software tool chain can be relied upon.

The required level of confidence in a software tool depends upon the circumstances of its deployment, with particular focus on the potential for a malfunctioning software tool to introduce or fail to detect errors in a safety-related development.

The LDRA tool suite has been qualified to ISO 26262 up to ASIL D, which removes considerable user overhead in providing evidence of that confidence.

Conclusions

SAE J3061 can be considered complementary to ISO 26262 in that it provides guidance on best development practices from a cybersecurity perspective, just as ISO 26262 provides guidance on practices to address functional safety.

SAE J3061 also calls for a similar sound development process to that of ISO 26262 and mirrors its development phases, and so the standards are not contradictory such that functional safety requirements are mirrored by cybersecurity requirements, designs must reflect both safety and cybersecurity, and so on.

Neither standard mandates the use of automated tools, but it is clear that their use is almost essential in all but the simplest of projects. Just as the standards complement each other, so the same tools can be used to support those standards concurrently.

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, Suite 228
Lewisville, Texas 75056
United States
Tel: +1 (855) 855 5372
e-mail: info@ldra.com

LDRA Technology Pvt. Ltd.
Unit No B-3, 3rd floor Tower B,
Golden Enclave. HAL Airport Road
Bengaluru
560017
India
Tel: +91 80 4080 8707
e-mail: india@ldra.com