LDRA
Software Technology

Technical Briefing

ISO 26262 a Pain in the ASIL?

Cost effective software certification in accordance with ISO 26262

www.ldra.com
* Registration required to download the document

© LDRA Ltd. This document is property of LDRA Ltd. Its contents cannot be reproduced, disclosed or utilised without company approval.

Background

There is an ever-widening range of automotive electrical and/or electronic (E/E/PE) systems such as adaptive driver assistance systems, anti-lock braking systems, steering and airbags. Their increasing levels of integration and connectivity provide almost as many challenges as their proliferation, with non-critical systems such as entertainment systems sharing the same communications infrastructure as steering, braking and control systems. The net result is a necessity for exacting functional safety development processes, from requirements specification, design, implementation, integration, verification, validation, and through to configuration.

ISO 26262 "Road vehicles – Functional safety" was published in response to this explosion in automotive E/E/PE system complexity, and the associated risks to public safety[1]. Like the rail, medical device and process industries before it, the automotive sector based their functional standard on the (largely) industry agnostic functional safety standard IEC 61508[2] which, in turn, drew heavily from the guiding principles of the aerospace standards such as DO-178B[3] /C[4] . The net result is that proven tools are available to help with the implementation of ISO 26262 which are longer established than the standard itself.

ISO 26262:2011 consists of 10 parts with three focused on product development: system level (Part 4)[5], hardware level (Part 5)[6], and software level (Part 6)[7]. It provides detailed industry specific guidelines for the production of all software for automotive systems and equipment, whether it is safety critical or not.

ISO 26262:2011 specifies a number of hazard classifications levels, known as ASILs (Automotive Safety Integrity Levels). ASILs range from A to D, so that the overhead involved in producing a safety critical ASIL D system (e. g. automatic braking) is greater than that required to produce an ASIL A system with few safety implications (e. g. the in-car entertainment system). ASILs are assigned as properties of each individual safety function, not as a property of the whole system or system component, and each assigned ASIL is influenced by the frequency of the situation ("exposure"), the potential impact should it occur ("severity"), and how easily it can then be managed ("controllability").

Security isn't explicitly identified as a consideration in ISO 26262, perhaps reflecting the fact that automotive embedded applications have traditionally been isolated, static, fixed function, device specific implementations, and practices and processes have relied on that status. Connection to the outside world changes things dramatically because it makes remote access possible while requiring no physical modification to the car's systems, most famously demonstrated in Miller and Valasek's work "Remote Exploitation of an Unaltered Passenger Vehicle"[8].

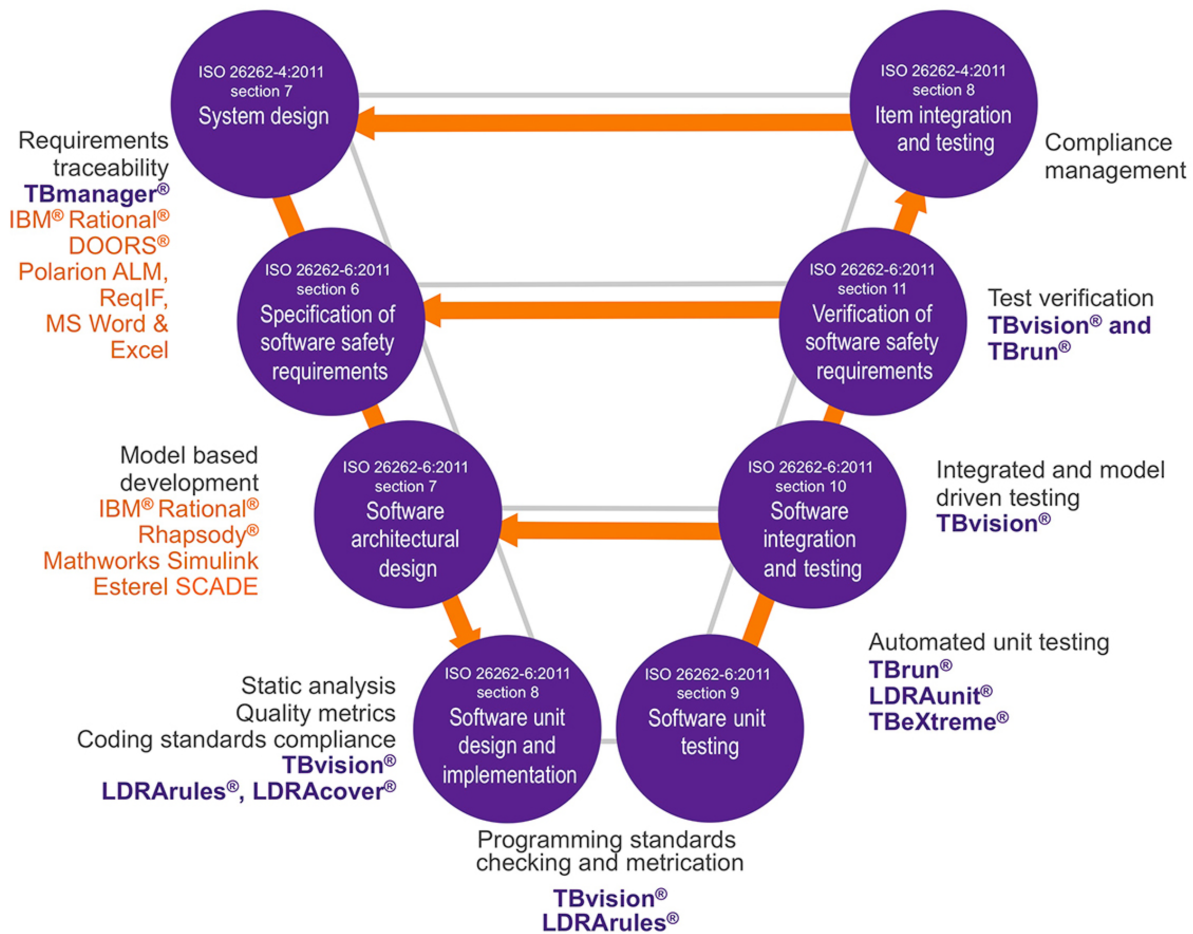
However, as for any other risk, as soon as there is potential for security vulnerabilities to threaten safety, ISO 26262 demands safety goals and requirements to deal with them. In short, the action to be taken to deal with each safety-threatening security issue needs to be proportionate to the risk (and hence ASIL).

ISO 26262 process objectives

A key element of ISO 26262-4:2011 is the practice of allocating technical safety requirements in the system design specification, and developing that design further to derive an item integration and testing plan. It applies to all aspects of the system including software, with the explicit subdivision of hardware and software development practices being dealt with further through the lifecycle.

[1] https://www.iso.org/news/2012/01/Ref1499.html
[2] IEC 61508:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems
[3] EUROCAE ED-12B December 1992 DO-178B/C, Software Considerations in Airborne Systems and Equipment Certification
[4] RTC DO-178C,2011,Software Considerations in Airborne Systems and Equipment Certification
[5] ISO 26262-4:2011 Road vehicles -- Functional safety -- Part 4: Product development at the system level
[6] ISO 26262-5:2011 Road vehicles -- Functional safety -- Part 5: Product development at the hardware level
[7] ISO 26262-6:2011 Road vehicles -- Functional safety -- Part 6: Product development at the software level
[8] http://illmatics.com/Remote%20Car%20Hacking.pdf Remote Exploitation of an Unaltered Passenger Vehicle, Dr. Charlie Miller & Chris Valasek, August 2015

The relationship between the system-wide ISO 26262-4:2011 and the software specific sub-phases found in ISO 26262-6:2011 can be represented in a V-model. Each of those steps is explained further in the following discussion.



System design (ISO 26262-4:2011 section 7)

The products of this system-wide design phase potentially include CAD drawings, spreadsheets, textual documents and many other artefacts, and clearly a variety of tools can be involved in their production. This phase also sees the technical safety requirements refined and allocated to hardware and software. Maintaining traceability between these requirements and the products of subsequent phases generally causes a project management headache.

The ideal tools for requirements management can range from a simple spreadsheet or Microsoft Word document to purpose-designed requirements management tool such as IBM Rational DOORS Next Generation[9] or Siemens Polarion PLM[10]. The selection of the appropriate tools will help in the maintenance of bi-directional traceability between phases of development, as discussed later.

Specification of software safety requirements (ISO 26262-6:2011 Section 6)

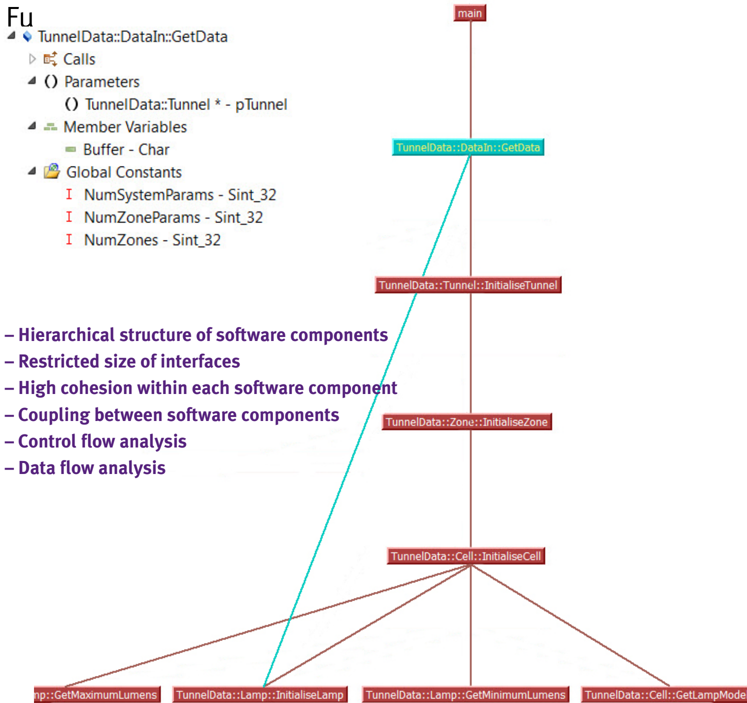
This sub-phase focuses on the specification of software safety requirements to support the subsequent design phases, bearing in mind any constraints imposed by the hardware.

It provides the interface between the product-wide system design of ISO 26262-4:2011 and the software specific ISO 26262-6:2011 and details the process of evolution of lower level, software related requirements. It will most likely involve the continued leveraging of the requirements management tools discussed in relation to the System Design sub-phase.

[9] http://www-03.ibm.com/software/products/en/ratidoor
[10] https://polarion.plm.automation.siemens.com/

Software architectural design (ISO 26262-6:2011 section 7)

There are many tools available for the generation of the software architectural design, with graphical representation of that design an increasingly popular approach. Appropriate tools are exemplified by MathWorks® Simulink®[11] , IBM® Rational® Rhapsody®[12] , and ANSYS® SCADE[13].



LDRA static analysis tools contribute to the verification of the design by means of the control and data flow analysis of the code derived from it, providing graphical representations of the relationship between code components for comparison with the intended design.

A similar approach can also be used to generate a graphical representation of legacy system code, providing a path for additions to it to be designed and proven in accordance with ISO 26262 principles.

Software unit design and implementation (ISO 26262-4:2011 section 8)

Coding rules: The illustration is a typical example of a table from ISO 26262-6:2011. It shows the coding and modelling guidelines to be enforced during implementation, superimposed with an indication of where compliance can be confirmed by the LDRA tool suite.

These guidelines combine to make the resulting code more reliable, less prone to error, easier to test, and/or easier to maintain. Peer reviews represent a traditional approach to enforcing adherence to such guidelines, and while they still have an important part to play, automating the more tedious checks using the LDRA tool suite is far more efficient, less prone to error, repeatable, and demonstrable.

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

ISO 26262-6:2011 highlights the MISRA coding guidelines language subsets as an example of what could be used. There are many different sets of coding guidelines available, but it is entirely permissible to use an in-house set or to manipulate, adjust and add one of the standard set to make it more appropriate for a particular application. The LDRA tool suite matches this flexibility.

[11] https://uk.mathworks.com/products/simulink.html
[12] http://www-03.ibm.com/software/products/en/ratirhapfami
[13] http://www.ansys.com/products/embedded-software/ansys-scade-suite

Software architectural design and unit implementation:

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

As for the coding guidelines before them, the guidelines relating to software architectural design and unit implementation are founded on the notion that they make the resulting code more reliable, less

TunnelData::Cell::Cell			
Float/integer conversion without cast.	Required	435 S	MISRA-C++:2008 5-0-5
Float/integer conversion without cast. : (double and int): f	Required	435 S	MISRA-C++:2008 5-0-5
Float/integer conversion without cast. : (double and int): f < NumLampTypes	Required	435 S	MISRA-C++:2008 5-0-5
Pointer subtraction not addressing one array.	Required	438 S	MISRA-C++:2008 5-0-17
Cast to an unrelated type. : (double* to int*): (Sint_32 *) p_f	Required	554 S	MISRA-C++:2008 3-9-3,5-2-7
Casting operation on a pointer. : (double* to int*): (Sint_			MISRA-C++:2008 5-2-7
Use of C type cast.			MISRA-C++:2008 5-2-4
Casting operation to a pointer. : (double* to int*): (Sint_32			MISRA-C++:2008 5-2-7

Standards Violation

prone to error, easier to test and/or easier to maintain. For example, architectural guidelines include:

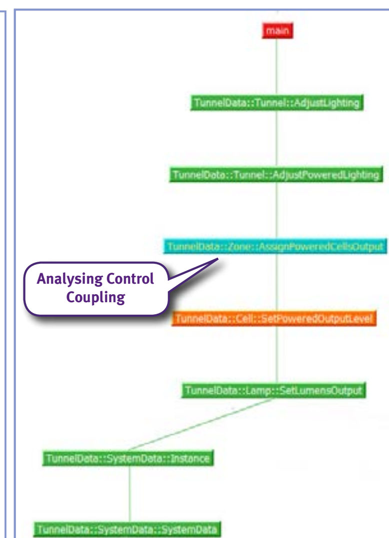
- Restricted size of software components and Restricted size of interfaces are recommended not least because large, rambling functions are difficult to read, maintain, and test – and hence more susceptible to error.
- High cohesion within each software component. High cohesion results from the close linking between the modules of a software program, which in turn impacts on how rapidly it can perform the different tasks assigned to it.

The LDRA tool suite provides metrics to ensure compliance with the standard such as complexity metrics as a product of interface analysis, cohesion metrics evaluated through data object analysis, and coupling metrics via data and control coupling analysis.

Requirement based test case

Unexecuted code for the given test case

Unexecuted data reference for the given test case



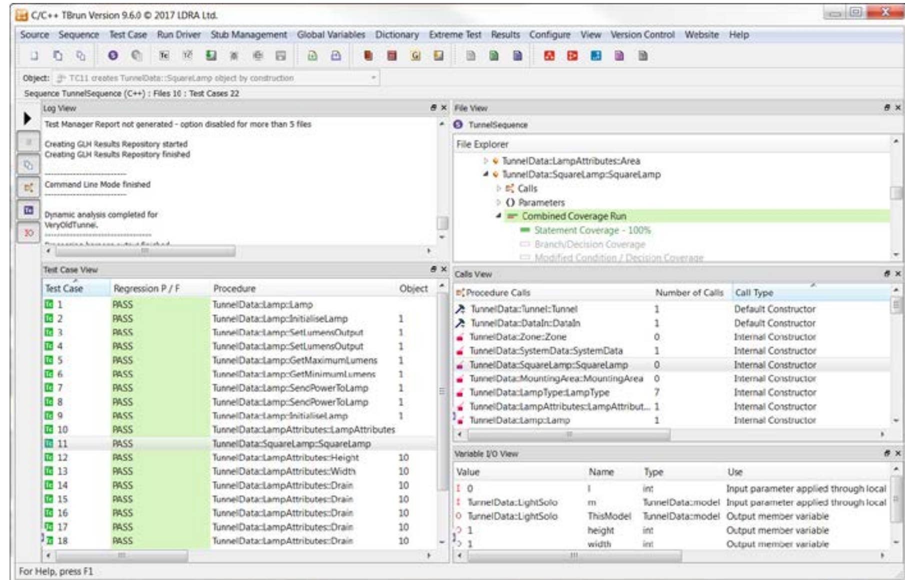
More generally, the LDRA tool suite can ensure that the good practices required by ISO 26262:2011 are adhered to whether they are coding rules, design principles, or principles for software architectural design.

In practice, for developers who are newcomers to ISO 26262, the role of the tool often evolves from a mechanism for highlighting violations, to a means to confirm that there are none.

Software unit testing (ISO 26262-6:2011 section 9) and Software integration and testing (ISO 26262-6:2011 section 10)

Just as static analysis techniques (an automated "inspection" of the source code) are applicable across the sub-phases of coding, architectural design and unit implementation, dynamic analysis techniques (involving the execution of some or all of the code) are applicable to unit, integration and system testing. Unit testing is designed to focus on particular software procedures or functions in isolation, whereas integration testing ensures that safety and functional requirements are met when units are working together in accordance with the software architectural design.

ISO 26262-6:2011 tables list techniques and metrics for performing unit and integration tests on target hardware to ensure that the safety and functional requirements are met and software interfaces are verified at the unit and integration levels. Fault injection and resource tests further prove robustness and resilience and, where applicable, back-to-back testing of model and code helps to prove the correct interpretation of the design. Artefacts associated with these techniques provide both reference for their management, and evidence of their completion. They include the software unit design specification, test procedures, verification plan and verification specification. On completing each test procedure, pass/fail results are reported and compliance with requirements verified appropriately.

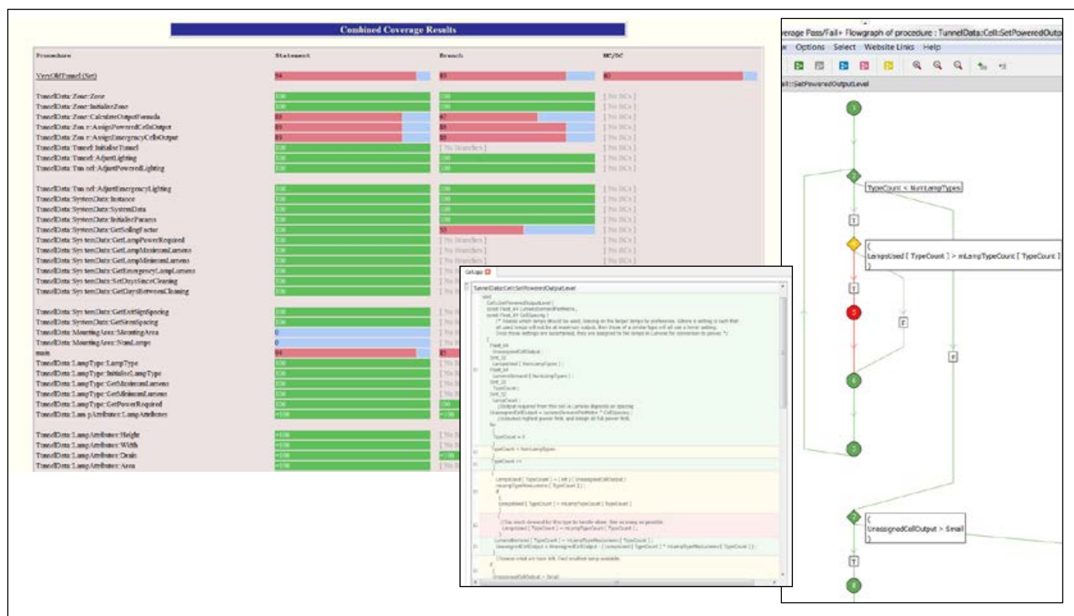


The example shows how the software interface is exposed at the function scope allowing the user to enter inputs and expected outputs to form the basis of a test harness The harness is then compiled and executed on the target hardware, and actual and expected outputs compared.

Unit tests become integration tests as units are introduced as part of a call tree, rather than being “stubbed”. Exactly the same test data can be used to validate the code in both cases.

The analysis of boundary values can be automated using an “extreme test” facility within the LDRA tool suite to automatically generate a series of unit test cases. The same facility also provides a facility for the definition of equivalence boundary values such as minimum value, value below lower partition value, lower partition value, upper partition value and value above upper partition boundary.

Should changes become necessary – perhaps as a result of a failed test, or in response to a requirement change from a customer - then all impacted unit and integration tests would need to be re-run (regression tested). The LDRA tool suite provides the means to automatically re-apply those tests to ensure that the changes do not compromise any established functionality.



ISO 26262:2011 does not require that any of the tests it promotes deploy software test tools. However, just as for static analysis, dynamic analysis tools help to make the test process far more efficient, especially for substantial projects.

Structural coverage metrics: In addition to showing that the software functions correctly, LDRA's dynamic analysis is used to generate structural coverage metrics. In conjunction with the coverage of requirements at the software unit level, these metrics provide the necessary data to evaluate the completeness of test cases and to demonstrate that there is no unintended functionality.

Metrics recommended by ISO 26262:2011 and provided by the LDRA tool suite include functional, call, statement, branch and MC/DC coverage. Unit and system test facilities can operate in tandem, so that (for instance) coverage data can be generated for most of the source code through a dynamic system test, and then be complemented using unit tests to exercise such as defensive constructs which are inaccessible during normal system operation.

Software test and model based development: The LDRA tool suite can be integrated with several different model based development tools, such as MathWorks Simulink, IBM Rational Rhapsody, and ANSYS SCADE. The development phase involves the creation of the model in the usual way, with the integration becoming more pertinent once source code has been auto generated from that model.

Using the MathWorks product as an example, "Back-to-back" testing is approached by first developing and verifying design models within Simulink. Code is then generated from Simulink, instrumented by the LDRA tool suite, executed in either Software in the Loop (SIL or host) mode, or Processor In the Loop (PIL or target) mode. Structural coverage reports are presented at the source code level by Simulink and the LDRA tool suite in tandem.

In addition to "back-to-back" testing, such an integration provides facilities to ensure that generated source code complies compliance with an appropriate coding standard, such as MISRA AC ACG[14], perform addition dynamic testing at the source code level, verify compliance with requirements, and test any hand-written additions to the auto generated code.

Bi-directional traceability (ISO 26262-4:2011 and ISO 26262-6:2011)

Bi-directional traceability runs as a principle throughout ISO26262:2011, with each development phase required to accurately reflect the one before it. In theory, if the exact sequence of the V-model is adhered to, then the requirements will never change and tests will never throw up a problem. But life's not like that.

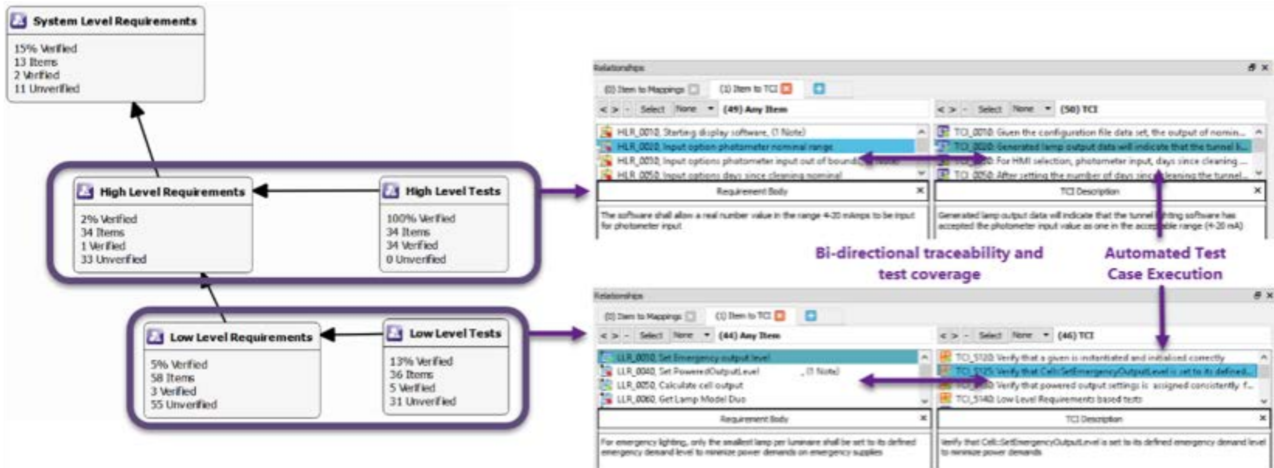
Consider, then, what happens if there is a code change in response to a failed integration test, perhaps because the requirements are inconsistent or there is a coding error. What other software units were dependent on the modified code?

Such scenarios can quickly lead to situations where the traceability between the products of software development falls down. Once again, while it is possible to maintaining traceability manually, automation helps a great deal.

Software unit design can take many forms – perhaps in the form of a natural language detailed design document, or perhaps model based. Either way, these design elements need to be bi-directionally traceable to both software safety requirements and the software architecture. The software units must then be implemented as specified and then be traceable to their design specification.

The LDRA tool suite can be used to establish traceability policy between requirements and tests cases of different scopes, which allows test coverage to be assessed. The impact of failed test cases can be assessed and addressed, as can the impact in requirements changes and gaps in requirements coverage. And artefacts such as traceability matrices can be automatically generated to present evidence of compliance to ISO 26262:2011.

[14] https://www.misra.org.uk/tabid/72/Default.aspx



In practise, initial structural coverage is usually accrued as part of this holistic process from the execution of functional tests on instrumented code leaving unexecuted portions of code which require further analysis. That ultimately results in the addition or modification of test cases, changes to requirements, and/or the removal of dead code. Typically, an iterative sequence of review, correct and analyse ensures that design specifications are satisfied.

Confidence in the use of software tools (ISO 26262-8:2011 section 11)

This supporting process defines a mechanism to provide evidence that the software tool chain is competent for the job. The required level of confidence in a software tool depends upon the circumstances of its deployment, both in terms of the possibility that a malfunctioning software tool can introduce or fail to detect errors in a safety-related element being developed, and the likelihood that such errors can be prevented or detected.

ZERTIFIKAT ◆ CERTIFICATE ◆ 認證證書 ◆ СЕРТИФИКАТ ◆ CERTIFICADO ◆ CERTIFICAT

Product Service

CERTIFICATE

No. Z10 16 09 84753 003

Holder of Certificate: LDRA Ltd.
Portside, Monks Ferry
Wirral
Merseyside CH41 5LH
UNITED KINGDOM

Factory(ies): 84753

Certification Mark:

Product: Software Tool for Safety Related Development

Model(s): LDRA tool suite
LDRArules
LDRAcover
LDRAunit
LDRAlite

Parameters: The certified tools, classified T2, fulfil the requirements for support tools according to IEC 61508-3 and EN 50128. The tools are qualified to be used in safety-related software development according to IEC 61508, EN 50128 and ISO 26262. It is suitably validated for use in safety-related development according to IEC 62304(ed.1)

The test report is a mandatory part of this certificate.

Tested according to: IEC 61508-3(ed.2)
ISO 26262-8:2011
EN 50128:2011

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. See also notes overleaf.

Test report no.: LW85043C

Valid until: 2021-10-03

Date, 2016-10-04

(Guido Neumann)

Page 1 of 1

TÜV SÜD Product Service GmbH · Zertifizierstelle · Ridlerstraße 65 · 80339 München · Germany

SGS TÜV SAAR

CERTIFICATE NO FS/71/220/15/0105 PAGE 1/1

LICENCE HOLDER
LDRA LTD
PORTSIDE, MONKS FERRY,
WIRRAL, CH41 5LH,
UNITED KINGDOM

MANUFACTURING PLANT
LDRA LTD
PORTSIDE, MONKS FERRY,
WIRRAL, CH41 5LH,
UNITED KINGDOM

PROJECT NO/-ID	LICENSED TEST MARK	CERT. REPORT NO.
K1C2-AU01	FUNKTIONALE SICHERHEIT GEPRÜFT / FUNCTIONAL SAFETY APPROVED	K1C20003

Tested according to: IEC 61508:2010 Ed. 2.0
ISO 26262:2011
EN 50128:2011
IEC 60880:2006
IEC 62304:2006

Certified product(s): LDRA tool suite
LDRAunit
LDRArules
LDRAcover
LDRAlite

Model(s): Version 9.5

Technical Data and Parameter: Usable in development of safety related software acc. to:
- ISO 26262 up to ASIL D, TCL1 can be reached
- IEC 61508 up to SIL 4, class T2 tool
- EN 50128 up to SW-SIL 4, class T2 tool
- IEC 62304 up to SW safety Class C
- IEC 60880

Specific Requirements: The certificate is based on voluntarily tests. Any changes to the design, components or processing may require repetition of some parts of the qualification in order to retain the certification.
The certificate report is an integral part of this certificate.
The safety related application guidelines (refer to K1C20003) shall be maintained.

Certification Body
for Functional Safety
SGS-TÜV Saar GmbH

Munich, 04.11.2015

Gudrun Neumann

CERTIFICATE

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

Depending on the user's assessment of their application, the LDRA tool suite will be assigned a "Tool Confidence Level" of either TCL1 or TCL2. In all cases except where the tool suite is assigned TCL2 and the product is designated ASIL D, the existence of a TUV certificate is sufficient to establish sufficient confidence in the tool. Otherwise, the tool is required to be subjected to a validation process, to show that the tool is capable of analysing sample software in the appropriate target environment.

A Tool Qualification Support Package (TQSP) is available from LDRA to provide that sample software.

www.ldra.com

LDRA

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

ISO 26262 a Pain in the ASIL v2.0 06/17