Reduce Cost of ISO 26262 Compliance while Driving Productivity Gains

M. Richardson

Reduce Cost of ISO 26262 Compliance while Driving Productivity Gains

Author: M.W.Richardson (mark.richardson@ldra.com)

Contents

Introduction

In response to the increased use of electronic systems within the automotive industry and particularly in recognition of their application to safety critical functions, the ISO 26262 standard has been created in order to comply with needs specific to the application sector of electrical / electronic / programmable electronic (E/E/PE) systems within road vehicles.

It is an adaptation of the IEC 61508 standard which was designed for use as the foundation for other industry specific standards. Previous examples of such adaptations include the CENELEC EN 50128 standard in the rail industry and the IEC 61511 standard in the process industry.
It also has much in common with the DO-178B standard seen in aerospace applications, particularly with respect to the requirement for MC/DC (Modified Condition/Decision Coverage) and the structural coverage analysis process.

The standard provides detailed industry specific guidelines for the production of all software for automotive systems and equipment, whether it is safety critical or not. It provides a risk-management approach including the determination of risk classes (Automotive Safety Integrity Levels, ASILs).

These ASILs are similar in nature to the SILs (Safety Integrity Levels) specified in the IEC 61508 standard. There are four levels of ASILs (A-D) to specify the necessary safety measures for avoiding an unreasonable residual risk, with ASIL D representing the most stringent level.
The ASIL is a property of a given safety function, not a property of the whole system or a system component. It follows that each safety function in a safety-related system needs to have an appropriate ASIL assigned with the risk of each hazardous event being evaluated based on the following attributes:

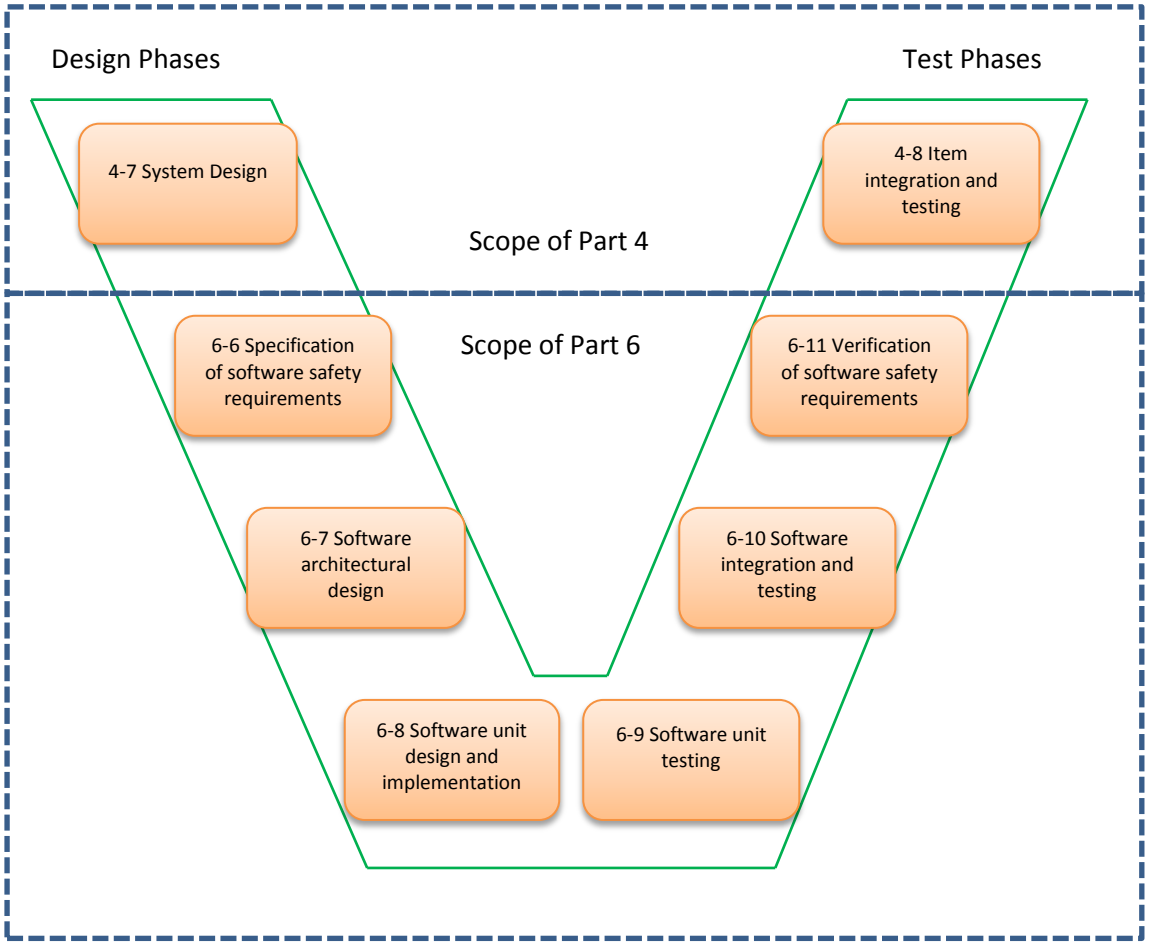
- Frequency of the situation (or “exposure”)
- Impact of possible damage (or “severity”)
- Controllability

Depending on the values of these three attributes, the appropriate automotive safety integrity level for a given functional defect is evaluated. This determines the overall ASIL for a given safety function. ISO 26262 translates these safety levels into safety specific objectives that must be satisfied during the development process. An assigned ASIL therefore determines the level of effort required to show compliance with the standard. This means that the effort and expense of producing a system critical to the continued safe operation of an automobile (e.g. a steer-by-wire system) is necessarily higher

than that required to produce a system with only a minor impact in the case of a failure (e.g. the in-car entertainment system).

ISO 26262 which was released in November 2011, demands a mature development environment that focuses on requirements that are specified in this standard. In order to claim compliance to ISO 26262, each requirement needs to be formally verified (There are possibly some exceptions where the requirement does not apply or where non-compliance is acceptable).

Part 4 of the standard, concerns the product development at the system level and part 6 of the standard, concerns the product development at the software level. The following diagram shows how the scope of these documents maps onto the familiar "V" model.



Part 6 of ISO 26262 "Product Development at the software level" covers the complete software lifecycle: planning, development and integral processes to ensure correctness, control and confidence in the software. These integral processes include requirements traceability, software design, implementation and verification.

Traceability (or Requirements Traceability) refers to the ability to link system requirements to software safety requirements, and then from software safety requirements to design elements and then to source code and the associated test cases. Although traceability is not explicitly identified as a requirement in the main body of the text, it is certainly desirable in ensuring the verifiability deemed necessary in section 7.4.2. Moreover, the need for "bi-directional traceability" (or upstream / downstream traceability) is noted in the same section (this would be very difficult to meet without using an automated traceability tool, such as offered by LDRA).

Structural Coverage (which utilises code coverage metrics) refers to the degree to which the source code of a system has been executed during requirements-based testing. Through the use of these practices it is possible to ensure that code has been implemented to address every system requirement and that the implemented code has been tested to completeness.

Verification tasks

Part 6 of the ISO 26262 standard contains a number of tables describing various verification tasks that are recommended for each ASIL, for example Table 1 in section 5.4.7

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

This section recommends that design and coding guidelines are used and as an example cites the use of the MISRA C or MISRA AC programming standards. Table 1 lists a number of topics that are to be covered by modelling and design guidelines. For example the enforcement of low complexity is highly recommended for all ASILs.

Not only does the LDRA tool suite cover all of the obligatory elements for each ASIL, but it also has the flexibility in configuration to allow less critical code in the same project to be associated with less demanding standards.

That principle extends to mixed C and C++ code, where appropriate standards are assigned to each file in accordance with its extension.

Another table in Part 6 is the Table 12 from section 9.4.5:

Topics		ASIL			
		A	B	C	D
1a	Statement coverage	++ ✓	++ ✓	+ ✓	+ ✓
1b	Branch coverage	+ ✓	++ ✓	++ ✓	++ ✓
1c	MC/DC (Modified Condition/Decision Coverage)	+ ✓	+ ✓	+ ✓	++ ✓

"++" 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

This table shows for instance that measuring statement coverage is highly recommended for all ASILs and that branch coverage is recommended for ASIL A and highly recommended for the other ASILs. For the highest ASIL D, MC/DC is also highly recommended.

Statement, branch and MC/DC coverage are provided by both the unit test and system test facilities within the LDRA tool suite. These packages 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 that can be complemented using unit tests to exercise defensive code and other aspects which are inaccessible during normal system operation.

Similarly table 15 in section 10.4.6 shows the structural coverage metrics at the software architectural level:

Topics		ASIL			
		A	B	C	D
1a	Function coverage	+ ✓	+ ✓	++ ✓	++ ✓
1b	Call coverage	+ ✓	+ ✓	++ ✓	++ ✓

"++" 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

Traceability

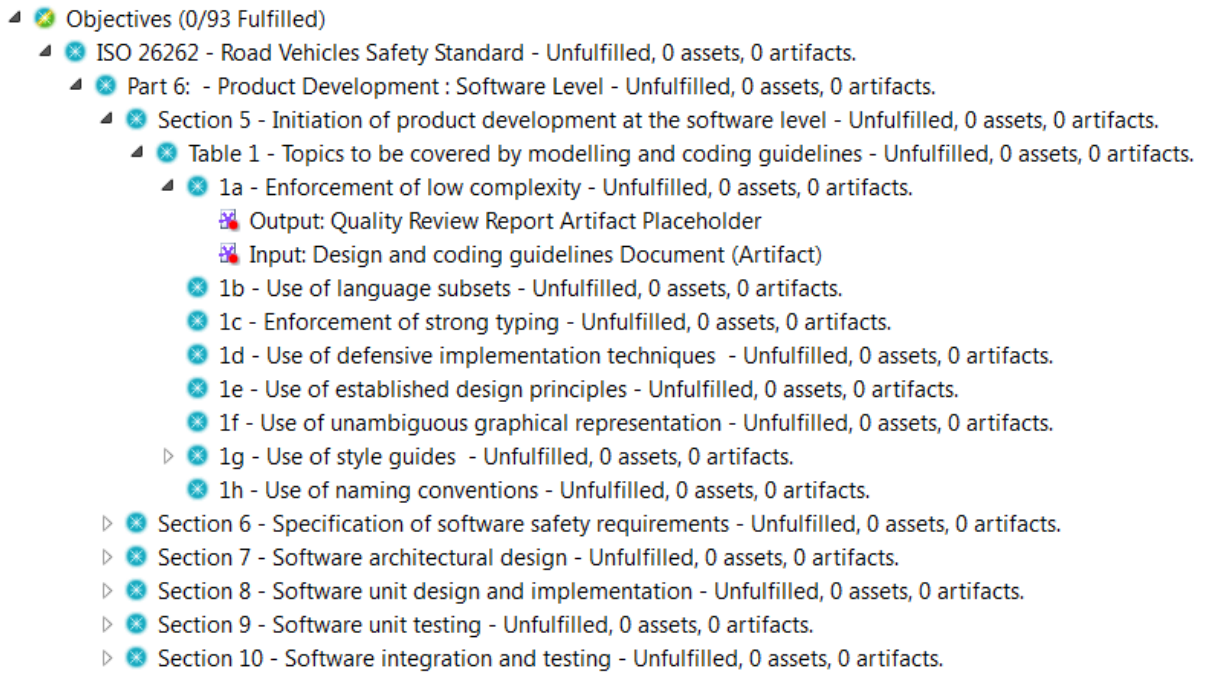
As we have seen there are two distinct phases; the design phase and then the verification and validation of the design, known as the test phase. From experience, the verification and validation of the design phase can take considerably more time to complete than the design phase.

One of the major challenges is firstly to maintain traceability between the system and software requirements down to the source code and test cases and secondly to maintain traceability from the various generated assets and artifacts back to the requirements specified in the ISO 26262 standard.

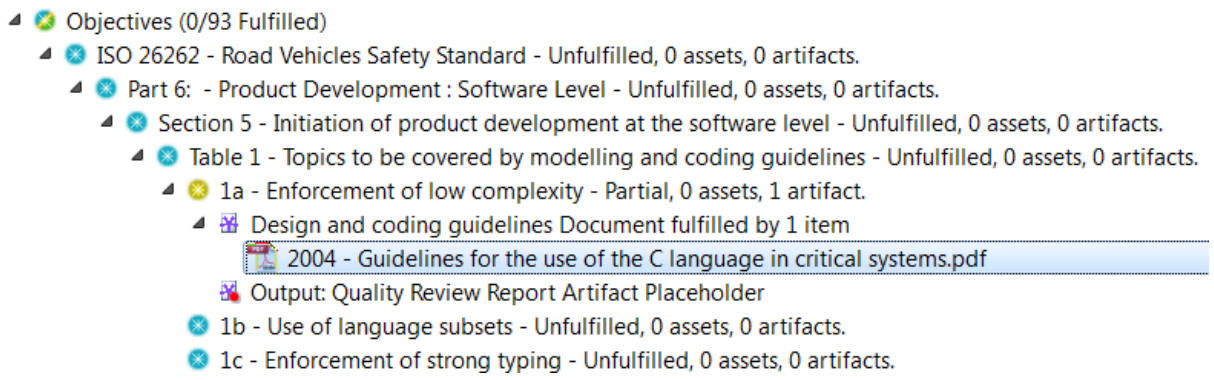
By *asset*, we mean items principally generated during the design phase such as: system requirements, software requirements, risk & safety documents, source code, etc. By *artifact* we mean items principally generated during the verification and validation of the design, such as: test cases, code coverage reports, code analysis reports, etc.

This is where the LDRA tool suite® can really reduce the cost of ISO 26262 compliance while providing significant productivity gains.

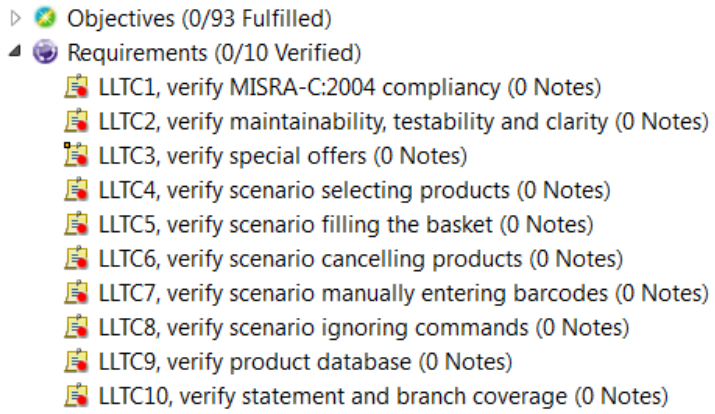
TBmanager® provides first of all the ability to capture the requirements specified by the ISO 26262 standard. In order to distinguish these *requirements* from the system and software *requirements*, TBmanager refers to them as *Objectives*. Associated with these Objectives are *placeholders* for the various assets and artifacts that need to be produced.



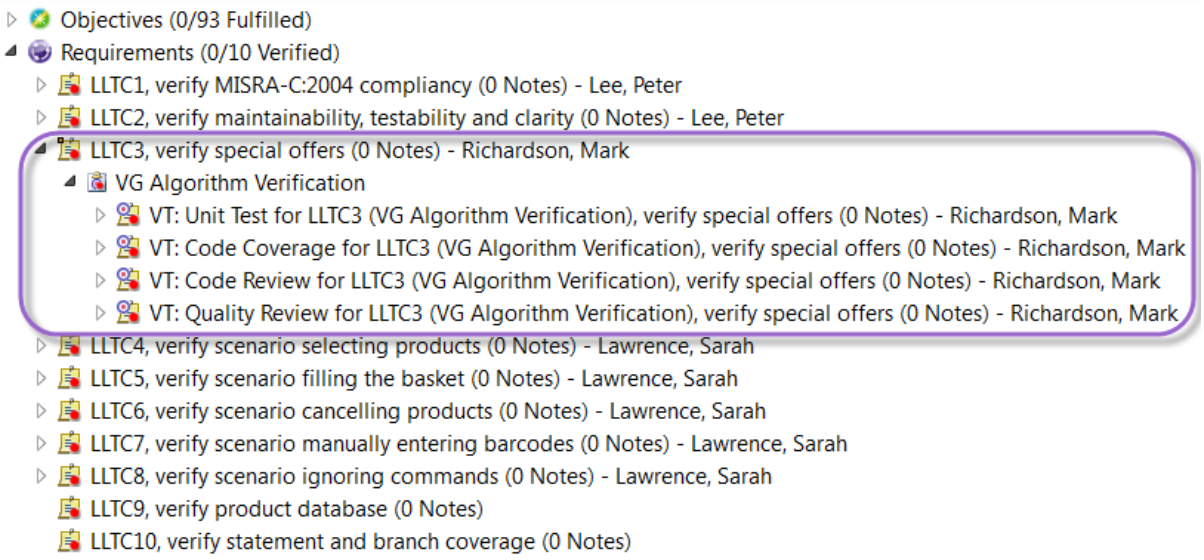
As these assets and artifacts get produced, they can be added to the project and then dragged and dropped onto the placeholders, thus providing traceability between these artifacts / assets and the ISO 26262 objectives. As assets / artifacts get linked to the objectives, then the status of the objective passes from unfulfilled to partial and eventually to fulfilled.



The system and software requirements can be imported using TBreq® into TBmanager from a wide range of different types of documents such as Microsoft Word®, Microsoft Excel® or IBM Rational DOORS®.

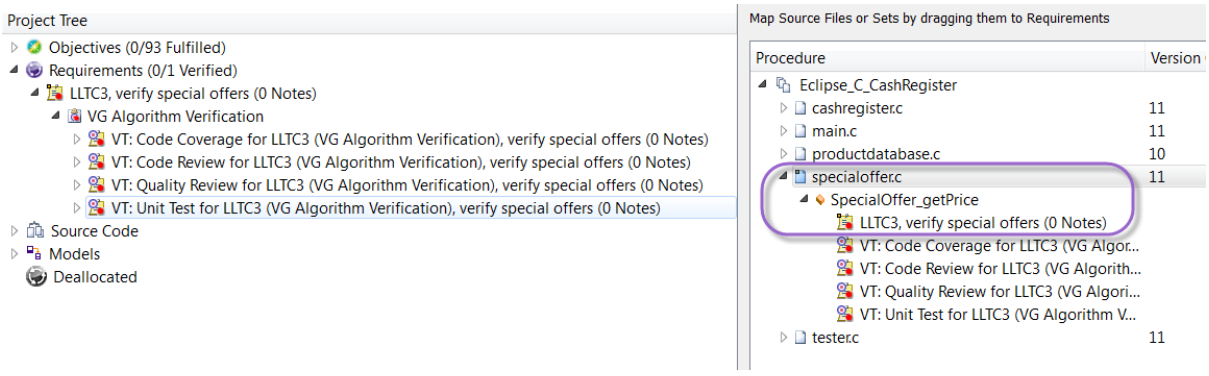


Once these requirements have been imported, then the project manager can assign appropriate verification tasks to each requirement, as well as assigning each requirement to a specific user.

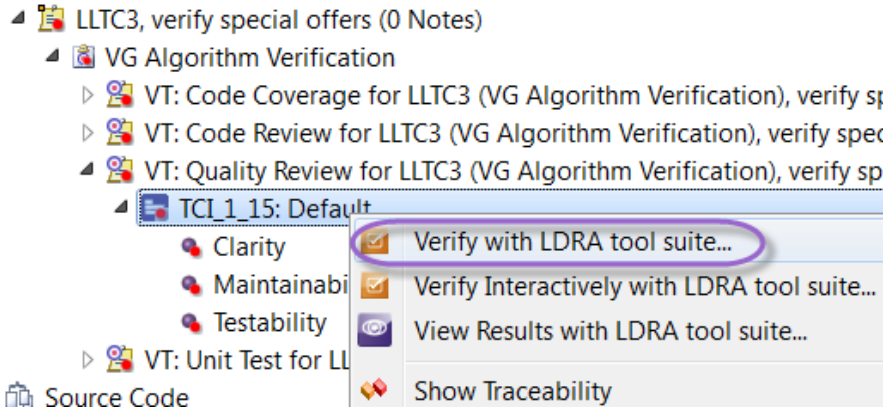


In this example, we can see that the requirement “LLTC3, verify special offers” has been assigned to a specific user and that the requirement has a number of verification tasks that need to be verified.

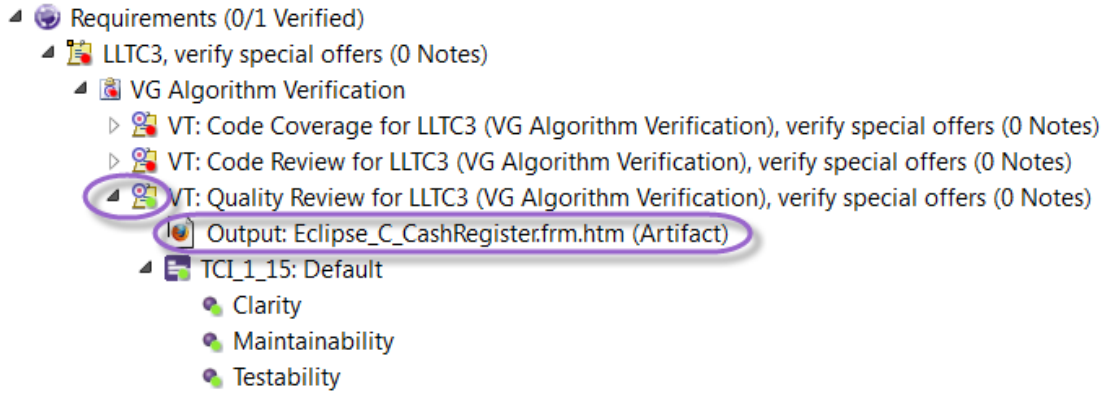
Automatically a TBmanager project will be created for that user and they will be able to see just the requirements that have been assigned to them. Their first task is to map the source code to the requirement.



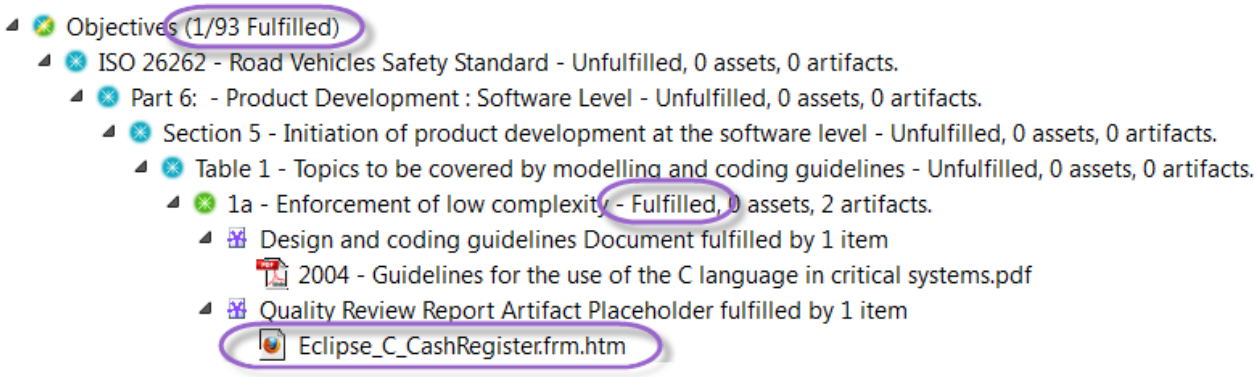
Once that is complete, then the verification tasks can be performed. For example a Quality Review can be automatically run using the LDRA tool suite.



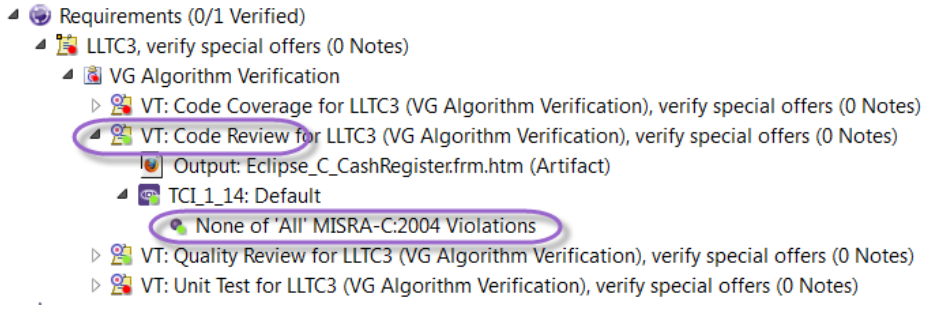
In this specific instance, a number of metrics (e.g. cyclomatic complexity, number of comments …) will be measured on the mapped source code and then verified to be within set limits, to ensure that the code is clear, maintainable and testable. The generated Quality Review report is automatically added as an artifact and if the metrics are all within the set limits, then the verification task is shown as verified.



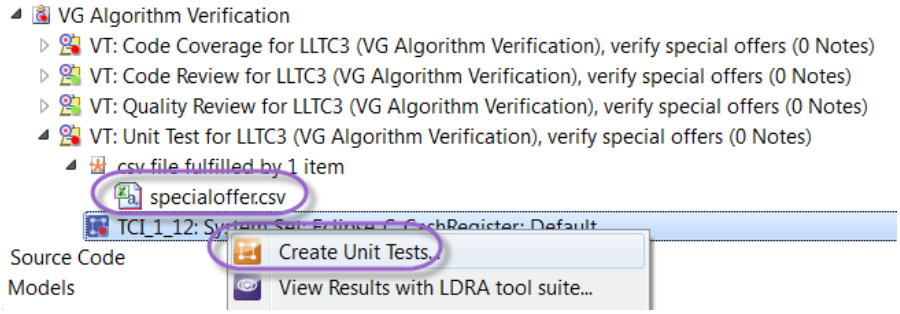
This artifact can now be used to satisfy one of the objective's placeholders:



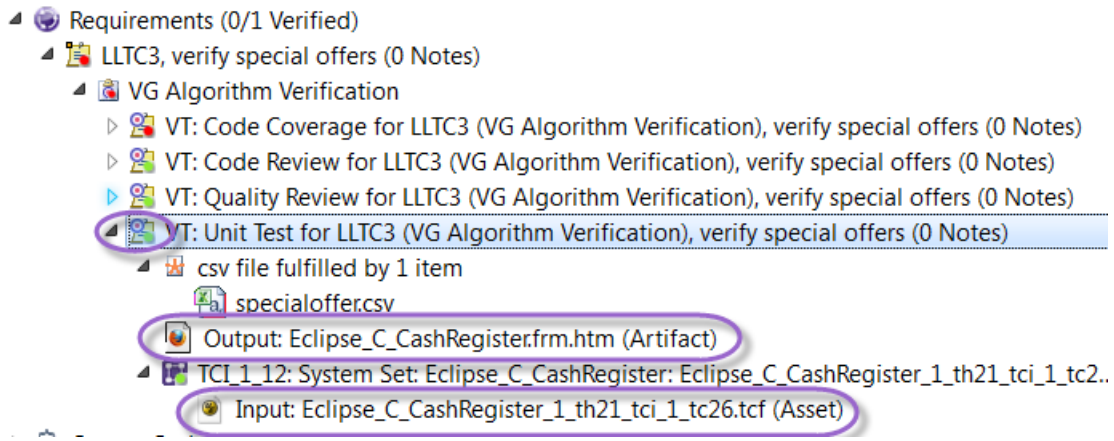
In a similar way, a Code Review can be automatically performed to ensure that the code meets the required programming standard, in this case MISRA-C:2004. Once more the generated report is added as an artifact that too can be used to satisfy a specific placeholder for an objective.



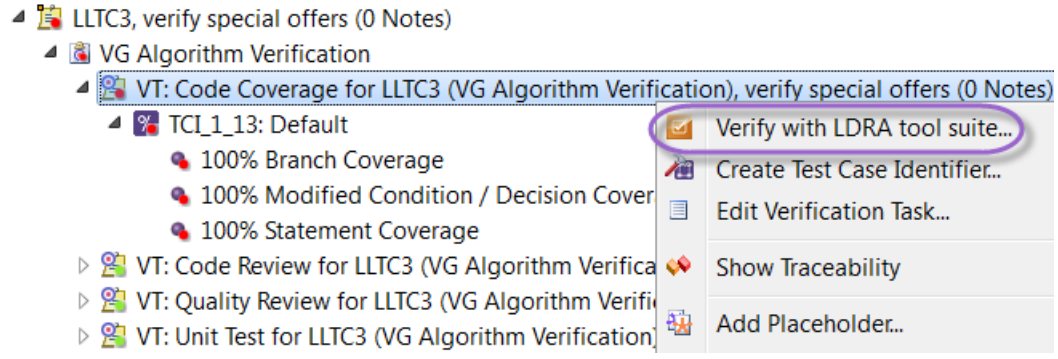
The next verification task is the unit test of the mapped source code. The input for creating the tests is in this specific case, a comma separated (.csv) file containing a list of inputs and expected outputs. This file can be added as an input artifact to the verification task.



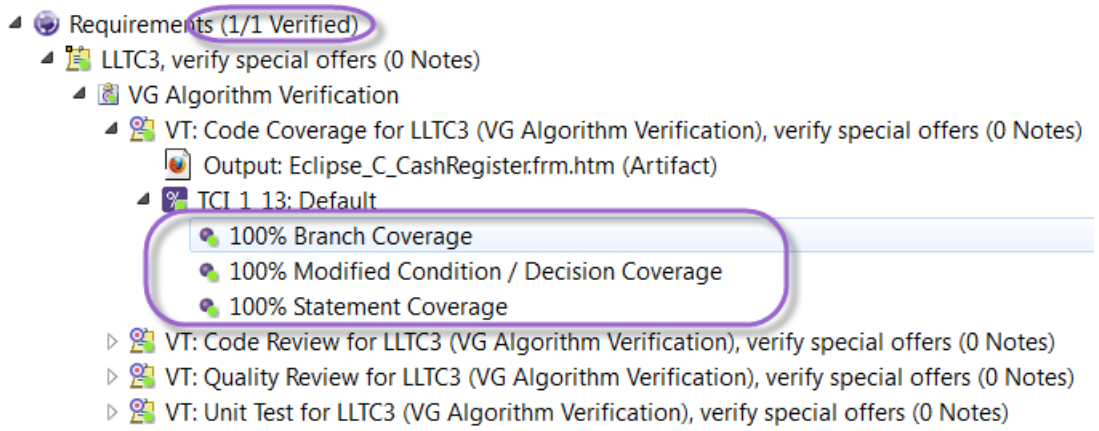
The unit testing can be done using the LDRA tool suite and once completed, the resulting test case file and test report are automatically added as artifacts to the verification task. Once more, these can be used to satisfy placeholders attached to objectives.



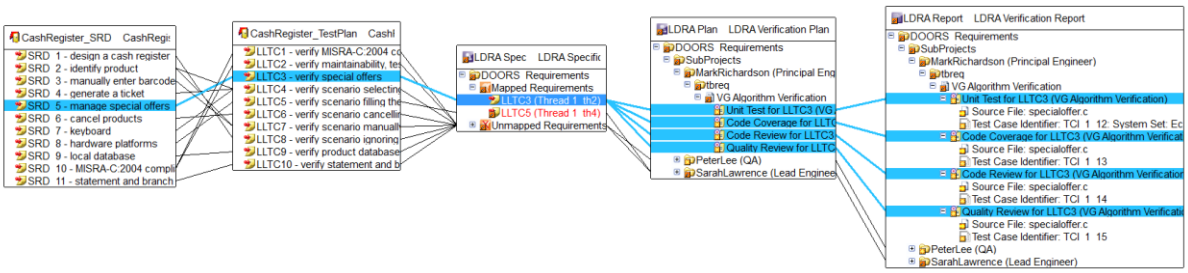
The final verification task is to verify the code coverage, for just the source code that was mapped to this requirement. This task can again be automatically executed using the LDRA tool suite:



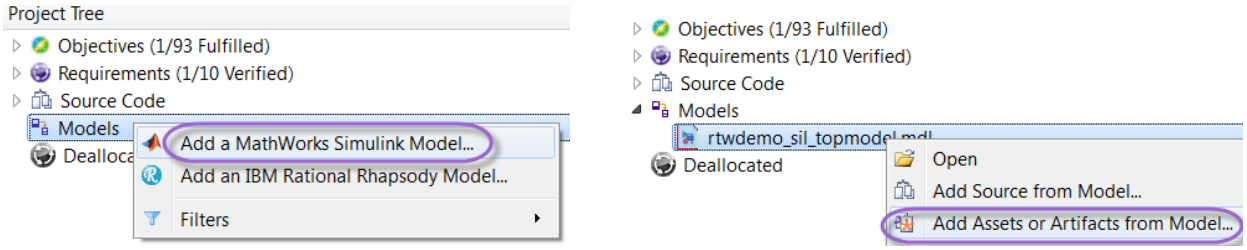
If the required branch, statement and MC/DC coverage is achieved, then the verification task passes and the requirement becomes verified.



Finally, TBreq can then be used to show the complete traceability from the high level requirements right down to the test cases and source code:



If code has been generated from a MathWorks Simulink® model, then TBmanager also allows that model to be added, as well as any corresponding source code, assets or artifacts from the model and then linked to the objectives.



Summary

As we have seen, through advanced traceability capability and integrated verification techniques, the LDRA tool suite can significantly help reduce the cost of meeting ISO 26262 compliance, while at the same time driving productivity gains.

For further information please visit: www.ldra.com