LDRA
Software Technology

Technical Briefing

Keeping EN 5012X compliant development on track with the LDRA tool suite®

www.ldra.com

Background

The approval process of interlocking systems mandates adherence to the CENELEC standards EN 50126-1 [1], EN 50126-2 [2], EN 50128 [3] and EN 50129 [4] across European countries. Collectively they describe the life cycle process for safety relevant Guided Transport Systems (GTS).

Like the automotive, medical device and process industries, the railway sector based their functional standard on the industry agnostic functional safety standard IEC 61508 [5]. The resulting EN 5012X series has become the dominant railway functional safety standard, and its requirements and processes are becoming increasingly familiar across the world. The international standards IEC 62278 [6], IEC 62779 [7], and IEC 62280 [8] very largely mirror EN 50126, EN 50128 and EN 50129 respectively and can be considered to be identical in the context of this document.

This document outlines the key software development and verification process activities of the standards, and uses LDRA's tool suite to show how automation can assist in proving compliance in a cost-effective manner. Extracts from the EN 5012X series are shown in *italics*.

Software in the context of the EN 5012X standards

Each of the EN 5012X series of standards has an impact on the software development lifecycle.

- EN 50126 is relevant to software systems, but not specifically focused upon them. It defines the four terms Reliability, Availability, Maintainability and Safety (RAMS), and describes their interaction and their management. It also defines a systematic process for specifying requirements for RAMS and demonstrating that those requirements are achieved.
- EN 50128 focuses specifically on software systems and their environment. It specifies procedures and technical requirements for the development of safety related programmable electronic systems for use in railway control and protection applications.
- EN 50126 is relevant to software systems, but not specifically focused upon them. EN 50129 specifies the lifecycle activities which are to be completed before the acceptance stage, and the activities to be carried out after it. It is primarily concerned with the evidence to be presented for the acceptance of safety-related systems.

Safety Integrity Levels (SILs)

EN 50129 specifies a number of hazard classifications known as Safety Integrity Levels (SILs). Development process checks and safety measures are specified to ensure a residual risk proportionate to the SIL. SILs range from Basic Integrity (formerly SIL 0) and then from 1 through to 4, where SIL 4 represents the most hazardous and hence demanding level.

TFFR per hour and per fonction	Safety Integrity Level
$10^{-9} \leq$ TFFR $< 10^{-8}$	4
$10^{-8} \leq$ TFFR $< 10^{-7}$	3
$10^{-7} \leq$ TFFR $< 10^{-6}$	2
$10^{-6} \leq$ TFFR $< 10^{-5}$	1

Figure 1: Table A.1 The SIL table, source: EN 50129:2018

EN 50129 discusses the derivation and assignment of SILs at some length. In brief, a SIL can be assigned to any safety-related system, sub-system or component performing a safety relevant function. The process starts with the identification of potential hazardous events. Then a Tolerable Hazard Rate (THR) is assigned for each hazardous event that might occur as a result of malfunction or failure of function, expressed as a probability per unit of time and taking into account risk reduction measures designed to reduce the rate of occurrence. Each hazard is then associated with a functional failure of a function or set of functions, and the THR used to derive a Tolerable Function Failure Rate (TFFR), from which a SIL can be assigned.

From the specific perspective of software development, the SIL assigned to each software component has a considerable impact on its verification and validation activity and the overhead associated with it, as illustrated in Figure 2.

	TECHNIQUE/MEASURE	REF.	SIL				
			Basic Integrity	1	2	3	4
1	Boundary Value Analysis	D.4	–	R	R	HR	HR
2	Checklists	D.7	–	R	R	R	R
3	Control Flow Analysis	D.8	–	HR	HR	HR	HR
4	Data Flow Analysis	D.10	–	HR	HR	HR	HR
5	Error Guessing	D.20	–	R	R	R	R
6	Walkthroughs/Design Reviews	D.56	HR	HR	HR	HR	HR

M" The method is mandatory for this SIL.
"HR" The method is highly recommended for this SIL.
"R" The method is recommended for this SIL.
"-" The method has no recommendation for or against its usage for this SIL.

Supported by the LDRA tool suite

Verified by the LDRA tool suite

Figure 2: Mapping the capabilities of the LDRA tool suite to "Table A.19 — Static analysis" from EN 50128:2011+A2

Security in the context of the EN 5012X standards

EN 5012X makes no specific mention of cybersecurity – but as soon as there is potential for security vulnerabilities to threaten safety, EN 50128 demands that they are dealt with as for any other hazards. The action to be taken to deal with each safety-threatening security issue therefore needs to be proportionate to the risk (and hence SIL). In acknowledgement of that, IEC 62334-4-1 [9] is gaining increasing prominence in an industry looking for more focused cybersecurity related guidance. It defines secure development life-cycle requirements related to cybersecurity for products intended for use in Industrial Automation and Control Systems (IACS) which are sufficiently complementary to those of EN 5012X to be integrated into the same software development lifecycle.

EN 5012X process objectives

EN 50128 §5.3.2.1 requires that "*a lifecycle model for the development of software shall be selected*" but stops short of dictating which model that should be, requiring only that it is to be detailed in the and includes examples of appropriate models, one of which is a V-model (Figure 3).

The different elements are not intended to be viewed as isolated silos; indeed, EN 50126 requires the safety team to identify the safety-related functional requirements and the safety-integrity requirements, and that they follow all of the requirements throughout the whole of the cycle – so that requirements are traceable to architectural design and vice versa, that architectural design is similarly traceable to component design and implementation, and so on.

The bidirectional nature of this traceability reduces the risk of failure by ensuring that not only is all required functionality present and proven, but also that there is no surplus code.

Automating EN 5012X processes

Tools can be applied to the EN 5012X processes as illustrated in Figure 3. For each phase, EN 50128 identifies several items of documentation to be created, and automating traceability between these different artefacts can help to make the process easier to manage and less error prone.

System development phase (EN 50126)

Although EN 50128 is the primary document for software development, it needs to be considered in the context of the system design for the product as a whole which is the domain of EN 50126. In this early part of the development lifecycle requirements are handled within one development process whether they impact software, hardware, or both.

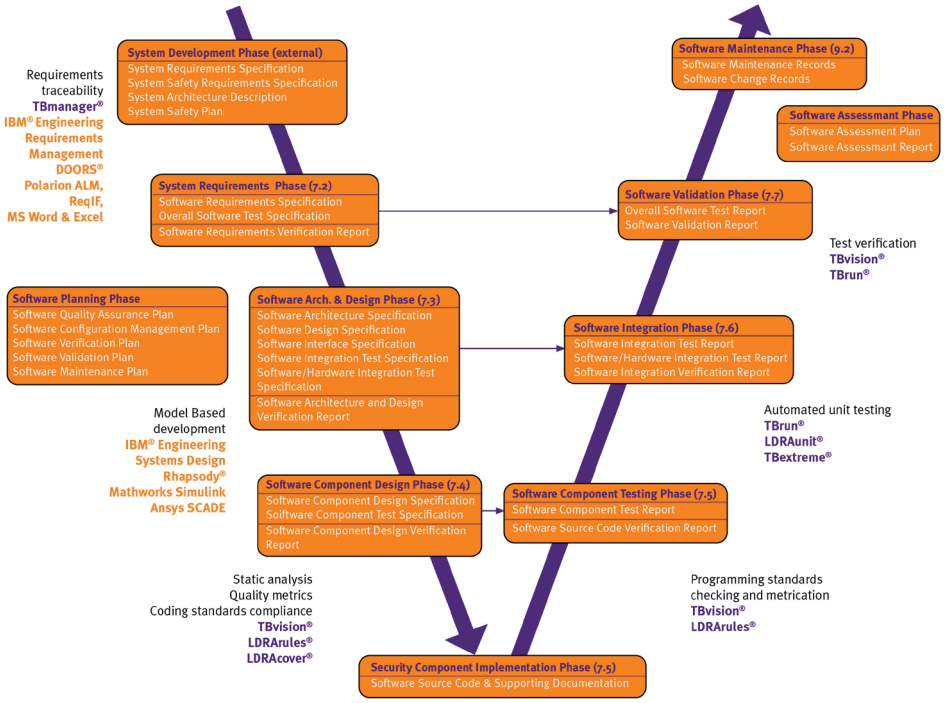

Figure 3: Mapping the capabilities of an automated tool chain to the V-model example of an appropriate EN 50128 lifecycle model

The ideal tools for requirements management depends largely on the scale of the development. If there are few developers in a local office, a simple spreadsheet or Microsoft Word document may suffice. Bigger projects, perhaps with contributors in geographically diverse locations, are likely to benefit from an Application Lifecycle Management (ALM) tool such as IBM Engineering Requirements Management DOORS [9], Siemens PLM Polarion ALM [10], or other ALM tools supporting the Requirements Interchange Format [11].

Software requirements phase (EN 50128 §7.2)

This phase details the process of evolution of lower level requirements as they relate specifically to the software system. That implies a continued leveraging of the requirements management tools selected for the project, as discussed in relation to the system development phase.

Software architecture and design phase (EN 50128 §7.3)

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 IBM Engineering Systems Design Rhapsody [12], MathWorks Simulink [13] and Ansys SCADE [14].

Table A.3 describes characteristics that are to be part of the architecture of the system. The TBmanager component of the LDRA tool suite could be used to show that such objectives are being fulfilled. Static analysis tools also have a part to play in the verification of the design in the form of control and data flow analysis of the code generated in accordance with it. As shown in Figure 4, the tools derive the relationship between some or all of the code components and represent it graphically such that it can then be compared with the intended design.

Software component design phase (EN 50128 §7.4)

The Software Component Design Specification is required to nominate techniques and measures from Table A.4, which references:

- EN 50128 §D.38 to describe a modular approach to limit the complexity of software.
- EN 50128 §D.2 which is concerned with ensuring that programs are easy for static analysis tools (including the LDRA tool suite) to cope with by adhering to the rules of structured programming.
- EN 50128 §D.53 which discusses the largely complementary concept of structured programming.

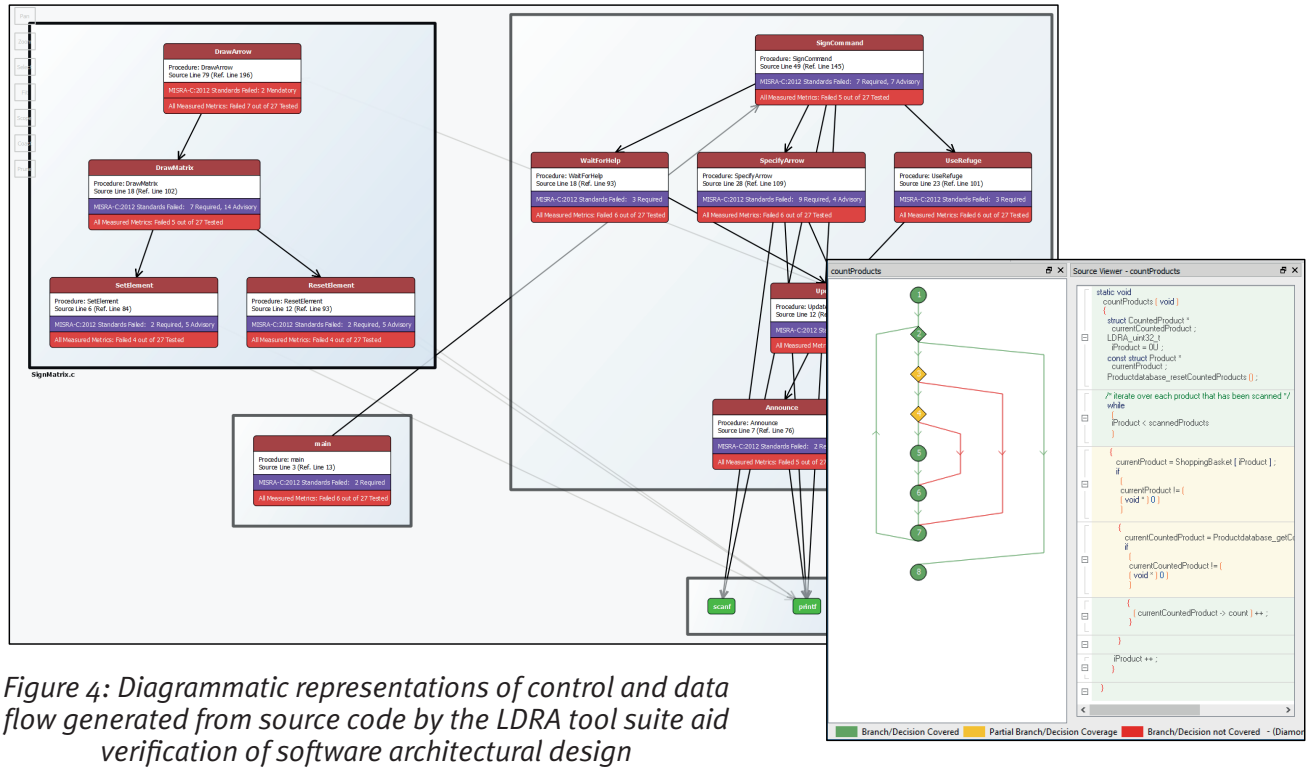

Figure 4: Diagrammatic representations of control and data flow generated from source code by the LDRA tool suite aid verification of software architectural design

The TBvision component of the LDRA tool suite supports these activities through quality metrics - measuring the complexity, loop nesting depth, procedure count, and other parameters with reference to configurable threshold values.

Table A.4 also references EN 50128 §D.35 and §A.12, which are concerned with “coding standards” and “coding rules”, and identifies specific language characteristics that are best avoided. The TBvision component of the LDRA tool suite can be used to verify adherence to the coding rules specified by the coding standard, style guide, and/or language subset. (Figure 5).

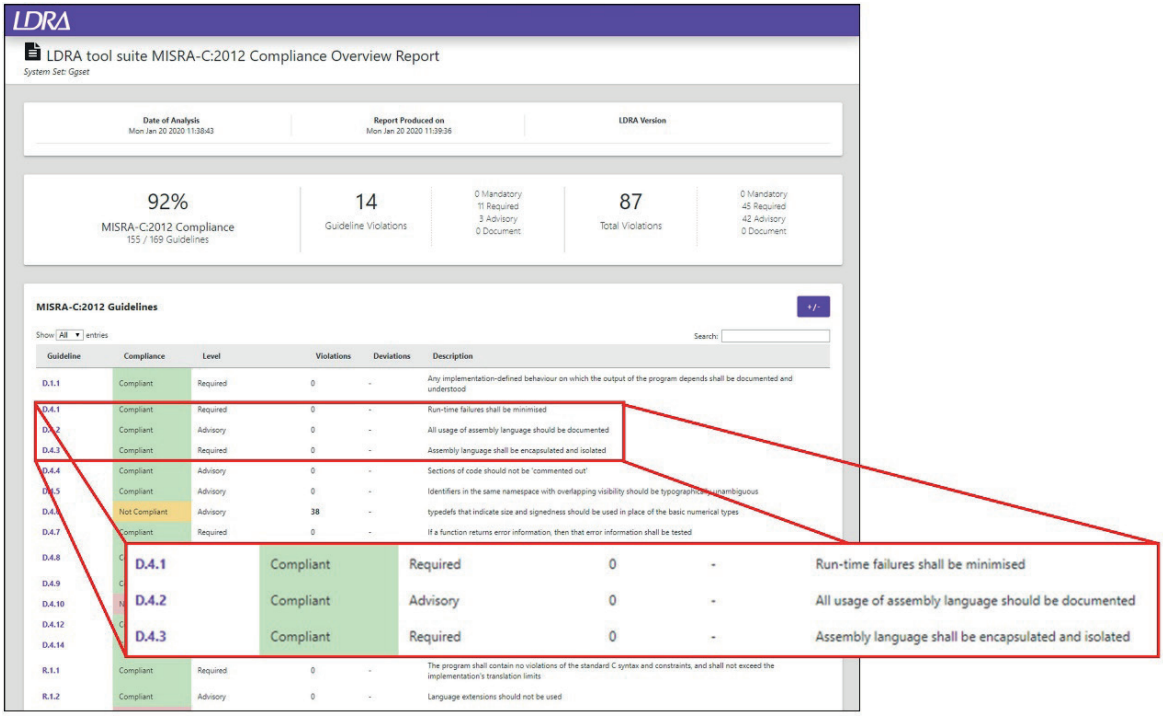

Figure 5: Reviewing coding rules and guidelines in the LDRA tool suite

Table A.5 in EN 50128 provides an overview of applicable verification and testing techniques, including static and dynamic analysis (sidebar). It references:

- EN 50128 §D.37 which discusses different quality metrics. The TBvision component of the LDRA tool suite can be used to calculate such quality metrics.
- EN 50128 §D.58 which concerns traceability. The TBmanager component of the LDRA tool suite is discussed in this regard later in this paper
- Boundary value analysis, which can be performed using the LDRA tool suite's static and dynamic analysis capabilities in combination.

Checklists are provided by the TBmanager component of the LDRA tool suite, and to some degree are automatically maintained by it.

Static analysis involves examining source code without executing the program it is part of.

Dynamic analysis involves executing programs, whether in part or as complete applications, on a real or virtual processor.

Both static and dynamic analysis can be undertaken with or without the assistance of automated tools.

The TBvision component offers the recommended control and data flow techniques described in Table A19, while the TBrun component supports the techniques highlighted in Table A.21. TBrun automatically generates test drivers and harnesses (wrapper code), enables tests to be easily and efficiently executed, and stores both test data and results. These tests can be automatically regressed. The test data maintenance process is streamlined through the automatic detection of changes in source code, prompting repeats of tests as necessary. The TBextreme option supplements TBrun, offering the option to automatically generate test cases, the nature of which is user configurable.

Table A.13 mentions boundary value analysis, used to detect software errors occurring at parameter limits or boundaries, and structure-based testing, which is closely related to the test coverage for code highlighted in Table A.5. The specified techniques can all be performed using the LDRA tool suite to determine the required coverage information as specified by the standard - statement, branch/decision, MC/ DC (Modified Coverage/Decision Coverage), etc.

Figure 6 illustrates some of the many devices used within the LDRA tool suite to illustrate that coverage.

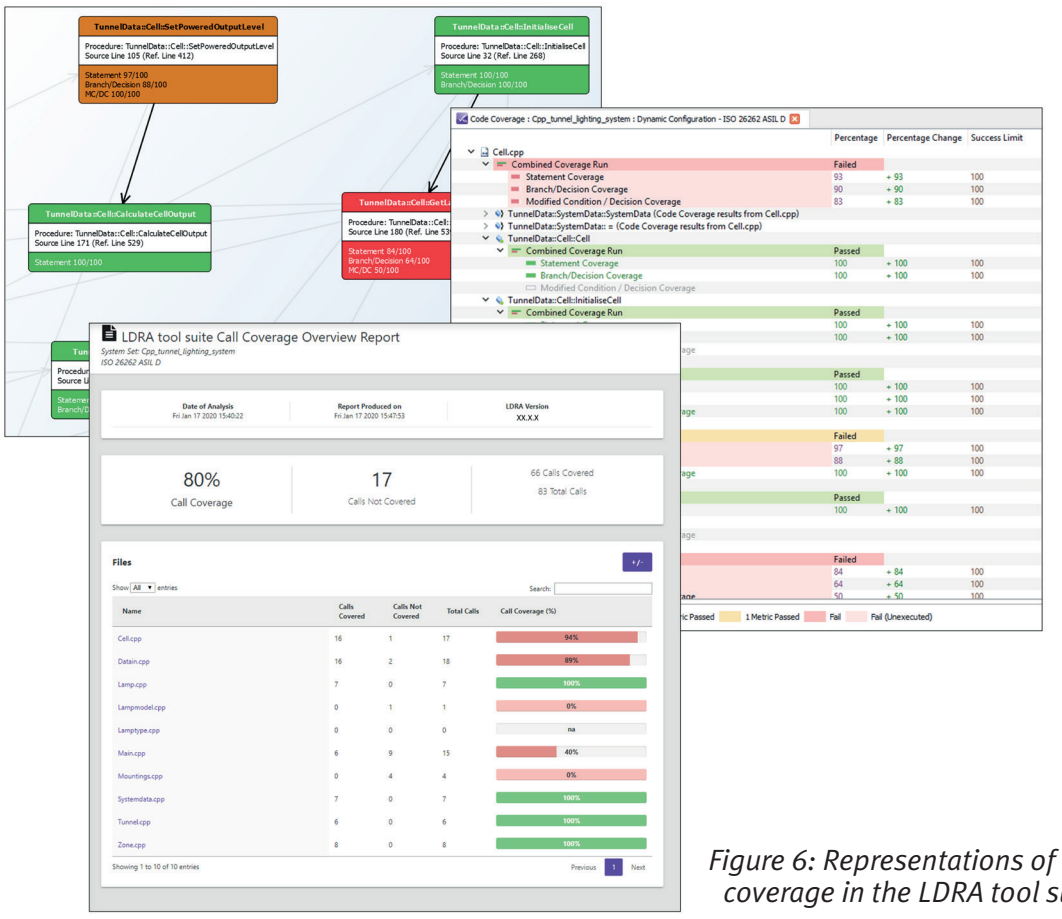

Figure 6: Representations of code coverage in the LDRA tool suite

Software component implementation and testing (EN 50128 §7.5)

Although this section of the standard represents a significant part of the development effort, it is quite short because it describes the implementation of the techniques that are specified at length earlier in the process. Most of the artefacts required by this section in EN 50128 §7.5.4.10 – including evidence of "*The correct use of the chosen techniques and measures from table A.4*", and "*the correct application of coding standards*" are generated by the tool suite as those measures are applied. The "*requirements for readability and traceability*" are supported by the TBmanager component of the LDRA tool suite, as discussed in the section "Managing the documentation" below.

Software integration phase (EN 50128 §7.6)

Much of the integration phase concerns the implementation of principles laid out in previous sections. The TBrun component of the LDRA tool suite supports software integration, testing on target hardware, timing analysis (to check whether the function is executed within a given period of time), and assembly code coverage on a multitude of different processors.

Software validation phase (EN 50128 §7.7)

Arguably the software validation phase is simply the last step in the integration phase, where all software and hardware components have been integrated into a complete system. It does however demand more complete tracing back to requirements and intermediate evidential artefacts, as reflected in the six input documents.

Managing the documentation

The EN 5012X series promotes the application of best practices along with the collation of evidence that those practices have been correctly applied. There are many evidential artefacts to collate, cross-reference and keep track of. Ensuring that these are permanently up-to-date can be a logistical nightmare in the face of changing customer requirements, failed tests and engineering oversights – all of which are real-life features of complex projects. Supporting that project management overhead by means of automation can make that process both more effective, and much less time consuming.

Software assessment phase (EN 50128 §6.4)

The software assessment phase puts into sharp focus the need to maintain evidential artefacts and their traceability at all times. It is shown detached from the V-model, and towards the end of it in parallel with the validation and maintenance phases, reflecting the intention for it to be a continuous process.

Bidirectional traceability

Traceability is referenced throughout the EN 5012X series of standards, and the repeated requirement for evidence of it (EN 50128 §5.3.2.7, §6.5.4.14, §7.5.4.10, §D.58 etc.) reinforces its role as a key objective of the standard.

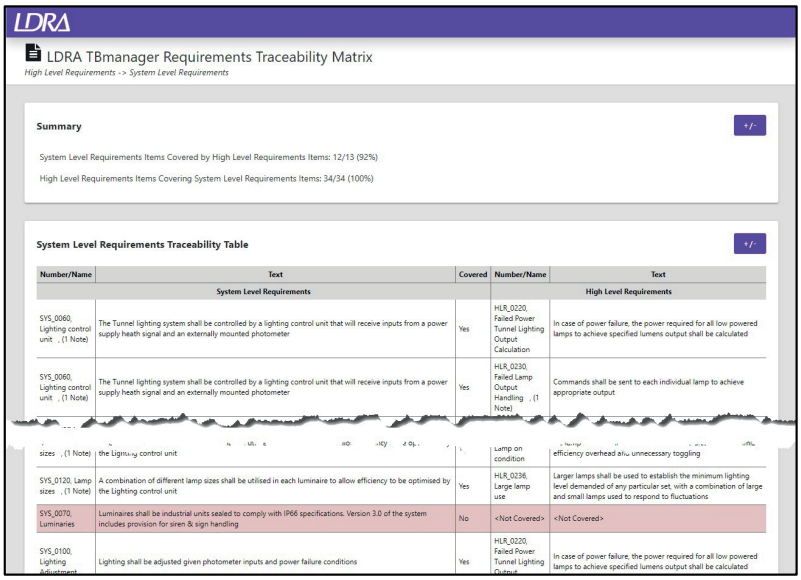

Figure 7: Highlighting missing coverage of requirements with the LDRA tool suite

It would be easy to assume that as long as the requirements are all shown to be implemented and tested in the code, then the objective of the standard is satisfied. However, it is just as important to show that there is no code present that satisfies no requirements. Such code may be the result of feature creep, of code that was created before an alternative implementation was selected, or even of a deliberately implanted "back door" method added by the unscrupulous. In any event, providing evidence that there is no such code is important.

Challenges arising during the course of a project mean that traceability between project phases can become challenging to achieve. Automating that traceability to both the project requirements and the standards' objectives and highlighting inconsistencies helps to address that challenge (Figure 7).

Software maintenance phase (EN 50128 §9.2)

Table A.10 covers software maintenance. It references impact analysis which aims to determine the extent to which a change or an enhancement to a software system will impact upon the existing system, and the resulting verification of the modified software system. Possible decisions are:

- Only the changed software module is re-verified
- All affected software modules are re-verified
- The complete system is re-verified

The analysis facilities provided by the TBmanager of the LDRA tool suite throughout the development lifecycle are arguably even more valuable during maintenance.

Support tools (EN 50128 §6.7)

The standard 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 reference to the possibility that the malfunctioning software tool and its corresponding erroneous output can introduce or fail to detect errors in a safety-related development, and the confidence in preventing or detecting such errors in its corresponding output.

Tools are categorized into one of three categories. The LDRA tool suite is placed in the T2 (Tool Class 2) category because it "*supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software*" (EN 50128 §3.1.42). That differentiates tools like the LDRA tool suite from (say) auto code generators which have the potential to introduce errors into an application.

Placement in that category requires that:

- "*Software tools shall be selected as a coherent part of the software development activities*" (EN 50128 §6.7.4.1)
- "*The selection of the tools in classes T2 and T3 shall be justified*" (EN 50128 §6.7.4.2)
- "*All tools in classes T2 and T3 shall have a specification or manual which clearly defines the behaviour of the tool and any instructions or constraints on its use*" (EN 50128 §6.7.4.3)
- "Configuration management shall ensure that for tools in classes T2 and T3, only justified versions are used" (EN 50128 §6.7.4.10)
- "*Each new version of a tool that is used shall be justified*" (EN 50128 §6.7.4.11)

The LDRA tool suite has been certified as "fulfilling the requirements of support tools according to EN 50128" by two separate TÜV organisations. The existence of such a certificate is sufficient to establish confidence in the tool, saving considerable overhead in establishing that independently.

Conclusions

The explosion in the quantity and complexity of the embedded software in Guided Transport Systems is well documented. The EN 5012X standards in their current form fine tunes the sound foundations established by the earlier versions, with the EN 50128 amendments bringing that standard into line with the rest of the set.

The EN 5012X standards do not directly address cybersecurity which is increasingly a concern for connected systems, and their use in parallel with the IEC 62334-4-1 standard is a popular approach to addressing that concern.

There is clearly an incentive to minimize the SIL applied to each element of an EN 5012X compliant system, because of the higher cost involved in achieving higher SILs. However, the whole principle of the assignment of SILs for various systems implies an assumption of separation, such that the most critical systems cannot be compromised by less critical functionality elsewhere. For a connected system to be considered safe and compliant with the principles of EN 5012X, it is therefore imperative that attack surfaces are minimized, and that separation between systems is optimized.

With that assurance in place, the task of maintaining the appropriate documentation at all times for each part of the system can be a daunting one. Automating not only the test processes but also the artefacts produced by them and other lifecycle activities can be of considerable help in alleviating the project management overhead of compliance. Tools such as those provided by LDRA are well proven in other safety critical sectors, making them not only ideally placed to further optimize the route to compliance, but also better established than the standard itself!

Works cited

[1] European Committee for Electrotechnical Standardization, EN 50126-1 “Railway Applications. The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS). Generic RAMS Process”, CENELEC, 2017.

[2] European Committee for Electrotechnical Standardization, EN 50126-2 “Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) - Systems Approach to Safety”, CENELEC, 2017.

[3] European Committee for Electrotechnical Standardization (CENELEC), EN 50128 “Railway application – Communication, signalling and processing systems - Software for railway control and protection systems”, CENELEC, 2020.

[4] European Committee for Electrotechnical Standardization, EN 50129 “Railway applications. Communication, signalling and processing systems. Safety related electronic systems for signalling”, CENELEC, 2017.

[5] International Electrotechnical Commission (IEC), IEC 61508:2010 “Functional safety of electrical/electronic/programmable electronic safety-related systems” (all parts), IEC, 2010.

[6] International Electrotechnical Commission (IEC), IEC 62278 “Railway applications - Specification and demonstration of reliability, availability, maintainability and safety (RAMS)”, IEC, 2002.

[7] International Electrotechnical Commission. (IEC), IEC 62279 “Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems”, IEC, 2015.

[8] International Electrotechnical Commission (IEC), IEC 62280 “Railway applications - Communication, signalling and processing systems - Safety related communication in transmission systems”, International Electrotechnical Commission IEC, 2014.

[9] International Electrotechnical Commission (IEC), IEC 64333-4-1 “Security for industrial automation and control systems -Secure product development lifecycle requirements”, IEC, 2018.

[10] IBM, “IBM Engineering Requirements Management DOORS Family,” [Online]. Available: https://www.ibm.com/uk-en/products/requirements-management [Accessed 20 January 2021].

[11] Siemens, “Polarion ALM,” [Online]. Available: https://polarion.plm.automation.siemens.com/products/polarion-alm [Accessed 20 January 2021].

[12] Object Management Group(R) , “Requirements Interchange Format,” Object Management Group(R), 2021. [Online]. Available: http://www.omg.org/spec/ReqIF/ [Accessed 10 August 2021].

[13] IBM, “IBM Engineering Systems Design Rhapsody,” [Online]. Available: https://www.ibm.com/uk-en/products/systems-design-rhapsody [Accessed 10 August 2021].

[14] The MathWorks, Inc., “Simulink Simulation and Model-Based Design,” MathWorks, 1994-2021. [Online]. Available: https://uk.mathworks.com/products/simulink.html [Accessed 10 August 2021].

[15] ANSYS, Inc, “Ansys SCADE Suite,” ANSYS, 2021. [Online]. Available: https://www.ansys.com/products/embedded-software/ansys-scade-suite [Accessed 10 August 2021].

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