Implementing EN 50128:2011 with the LDRA tool suite®

Railway Applications. Communication, Signalling and Processing Systems. Software for Railway Control and Protection Systems

www.ldra.com

* Registration required to download the document

Introduction

The EN 50128:2011 standard is part of a group of related standards:

- EN 50126-1:1999 - Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) - Part 1: Basic requirements and generic process.
- EN 50129:2003 - Railway applications - Safety related electronic systems for signaling. EN 50126 addresses system issues on the widest scale, while EN 50129 addresses the approval process for individual systems which may exist within the overall railway control and protection system.

The scope of EN 50126-1 and EN 50129 is to define the process of specifying the safety functions allocated to software.

The standard contains a number of tables:

- Annex A (normative) – Criteria for the selection of techniques and measures which consists of clause tables and detailed tables.
- Annex B (normative) – Key software roles and responsibilities
- Annex C (informative) – Documents control summary
- Annex D (informative) – Bibliography of techniques

This document illustrates how the LDRA tool suite can be applied to address the requirements of each table in Annex A. For each table, for which a contribution can be made, a brief description of the appropriate LDRA tool suite feature is provided. When choosing between fault finding strategies, it is worthwhile to recall that in general static analysis techniques find the technical faults (due to language defects, array bound faults, common programmer mistakes, divide–by-zero, etc.), and application faults are found by techniques exploiting knowledge of requirements, such as, dynamic analysis and other types of formal methods.

Safety Integrity Level (SIL):

The EN 50128 standard identifies techniques and measures for five levels of software safety integrity, where 0 is the minimum level and 4 the highest level. In this version (EN 50128:2011), techniques required for level 1 are the same as for level 2 and the techniques required for level 3 are the same as for level 4. These levels have been included as normative, in order to allow a smooth transition between software developments for non-safety related systems and for safety-related systems.

SOFTWARE SAFETY INTEGRITY LEVEL	DESCRIPTION OF SOFTWARE
4	Very High
3	High
2	Medium
1	Low
0	Non Safety Related

The required techniques and measures, for each software safety integrity level and for the non safety-related level, are shown in the annex tables.

Development Lifecycle:

The EN 50128 development lifecycle is defined in phases and activities. The phases of the lifecycle are listed below:

- System Development Phase (external)
  - System Requirement Specification
  - System Safety Requirement Specification
  - System Architecture Plan
  - System Safety Plan
- Software Requirement Phase (7.2)
  - Software Requirements Specification
  - Overall Software Test Specification
  - Software Requirement Verification Report
- Software Planning Phase
  - Software Quality Assurance Plan
  - Software Configuration Management Plan
  - Software Verification Plan
  - Software Validation Plan
  - Software Maintenance Plan
- Software Architecture & Design Phase (7.3)
  - Software Architecture Specification
  - Software Design Specification
  - Software Interface Specification
  - Software Integration Test Specification
  - Software/Hardware Integration Test Specification
  - Software Architecture & Design Verification Report
- Software Component Design Phase (7.4)
  - Software Component Design Specification
  - Software Component Test Specification
  - Software Component Design Verification Report
- Software Component Implementation Phase (7.5)
  - Software Source Code and Supporting Documentation
- Software Component Testing Phase (7.5)
  - Software Component Test Report
  - Software Source Code Verification Report
- Software Integration Phase (7.6)
  - Software Integration Test Report
  - Software/Hardware Integration Test Report
  - Software Integration Verification Report
- Software Validation Phase (7.7)
  - Overall Software Test Report
  - Software Validation Report
- Software Assessment Phase
  - Software Assessment Plan
  - Software Assessment Report

- Software Maintenance Phase (9.2)
    - o Software Maintenance Records
    - o Software Change Records

Software Assurance consists of the following phases:

- Software Testing (6.1)
- Software Verification (6.2)
- Software Validation (6.3)
- Software Assessment (6.4)
- Software Quality Assurance (6.5)
- Modification and Change Control (6.6)
- Support Tools and Languages (6.7)

Annex A Tables:

The annex tables are also referred to as clause tables. In the following, we reproduce each table and describe the relevant features of the LDRA tool suite. The rows in each table will be referred to as numbered items.

DOCUMENTATION		SWSIL				
		0	1	2	3	4
Planning						
1	Software Quality Assurance Plan	++	++	++	++	++
2	Software Quality Assurance Verification Report	++	++	++	++	++
3	Software Configuration Management Plan	++	++	++	++	++
4	Software Verification Plan	++	++	++	++	++
5	Software Validation Plan	++	++	++	++	++
Software Requirements						
6	Software Requirements Specification	++	++	++	++	++
7	Overall Software Test Specification	++	++	++	++	++
8	Software Requirements Verification Report	++	++	++	++	++
Architecture and Design						
9	Software Architecture Specification	++	++	++	++	++
10	Software Design Specification	++	++	++	++	++
11	Software Interface Specifications	++	++	++	++	++
12	Software Integration Test Specification	++	++	++	++	++
13	Software/Hardware Integration Test Specification	++	++	++	++	++
14	Software Architecture and Design Verification Report	++	++	++	++	++
Component Design						
15	Software Component Design Specification	+	++	++	++	++
16	Software Component Test Specification	+	++	++	++	++
17	Software Component Design Verification Report	+	++	++	++	++
Component Implementation and Testing						
18	Software Source Code and Supporting Documentation	++✔	++✔	++✔	++✔	++✔
19	Software Component Test Report	+✔	++✔	++✔	++✔	++✔

20	Software Source Code Verification Report	++✔	++✔	++✔	++✔	++✔
Integration						
21	Software Integration Test Report	++✔	++✔	++✔	++✔	++✔
22	Software/Hardware Integration Test Report	++✔	++✔	++✔	++✔	++✔
23	Software Integration Verification Report	++✔	++✔	++✔	++✔	++✔
Overall Software Testing / Final Validation						
24	Overall Software Test Report	++✔	++✔	++✔	++✔	++✔
25	Software Validation Report	++	++	++	++	++
26	Tools Validation Report	+	++	++	++	++
27	Release Note	++	++	++	++	++
Systems Configured by Application Data/ Algorithms						
28	Application Requirements Specification	++	++	++	++	++
29	Application Preparation Plan (see NOTE 2)	++	++	++	++	++
30	Application Test Specification (see NOTE 2)	++	++	++	++	++
31	Application Architecture and Design (see NOTE 2)	++	++	++	++	++
32	Application Preparation Verification Report	++	++	++	++	++
33	Application Test Report	++	++	++	++	++
34	Source Code of Application Data/Algorithms	++	++	++	++	++
35	Application Data/Algorithms Verification Report	++	++	++	++	++
Software Deployment						
36	Software Release and Deployment Plan	+	++	++	++	++
37	Software Deployment Manual	+	++	++	++	++
38	Release Notes	++	++	++	++	++
39	Deployment Records	+	++	++	++	++
40	Deployment Verification Report	+	++	++	++	++
Software Maintenance						
41	Software Maintenance Plan	+	++	++	++	++
42	Software Change Records	++	++	++	++	++
43	Software Maintenance Records	+	++	++	++	++
44	Software Maintenance Verification Report	+	++	++	++	++
Software Assessment						
45	Software Assessment Plan	+	++	++	++	++
46	Software Assessment Report	+	++	++	++	++

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.1 — Lifecycle issues and documentation (5.3)

Table A.1 refers to lifecycle issues and documentation. TBvision® generates software verification reports where the source will be checked against programming standards such as MISRA, CERT, DERA, etc. TBrun® generates test reports for unit, module or integration (software-software OR software-hardware) test showing

functionality results as well as structural coverage results. TBreq® produces an overall software verification report explaining traceability between requirements to the source code file and procedures, test case identifiers and a pass/fail status for a given requirement. TBreq's Requirements Traceability Matrix (RTM) report shows traceability starting from requirements through design documentation, source code and test cases to test results.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Formal Methods (based on a mathematical approach)	D.28	○	+✔	+✔	++✔	++✔
2	Modelling	Table A.17	+✔	+✔	+✔	++✔	++✔
3	Structured Methodology	D.52	+	+	+	++	++
4	Decision Tables	D.13	+	+	+	++	++

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.2 — Software requirements specification (7.2)

Table A.2 specifies software requirements specification. Formal methods (section D.28) transfer the principles of mathematical reasoning to the specification and implementation of technical systems. The static analysis features of the LDRA tool suite, analyse the source code with respect to various programming models (e.g. MISRA C), where the LDRA tool suite mathematically analyses the structure and semantics of the program to provide analysis reports. This analysis feature of the LDRA tool suite can be linked with TBreq (LDRA's requirement traceability tool), which provides bi-directional traceability between the software requirement specification and input documents (i.e. System Requirement Specification, System Safety Requirements Specification, System Architecture Description, External Interface Specifications, Software Quality Assurance Plan and Software Validation Plan) required for this phase.

Modelling can be used to specify the software requirements. The source code generated from the model should be verified against the requirements. The LDRA tool suite provides integration with different modelling tools such as IBM® Rational® Rhapsody®, MathWorks® Simulink®, Esterel® SCADE®, such that the source generated from these models is validated by performing functional/black-box testing and structural coverage analysis.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Defensive Programming	D.14	○	++✔	++✔	++✔	++✔
2	Fault Detection and Diagnosis	D.26	○	+✔	+✔	++✔	++✔
3	Error Correcting Codes	D.19	○	○	○	○	○
4	Error Detecting Codes	D.19	○	+✔	+✔	++✔	++✔
5	Failure Assertion Programming	D.24	○	+✔	+✔	++✔	++✔
6	Safety Bag Techniques	D.47	○	+	+	+	+
7	Diverse Programming	D.16	○	+	+	++	++
8	Recovery Block	D.44	○	+	+	+	+
9	Backward Recovery	D.5	○	○	○	○	○
10	Forward Recovery	D.30	○	○	○	○	○

11	Re-try Fault Recovery Mechanisms	D.46	○	+	+	+	+
12	Memorising Executed Cases	D.36	○	+✔	+✔	++✔	++✔
13	Artificial Intelligence - Fault Correction	D.1	○	○	○	○	○
14	Dynamic Reconfiguration of Software	D.17	○	○	○	○	○
15	Software Error Effect Analysis	D.25	○	+	+	++	++
16	Graceful Degradation	D.31	○	+	+	++	++
17	Information Hiding	D.33	○	○	○	○	○
18	Information Encapsulation	D.33	+	++	++	++	++
19	Fully Defined Interface	D.38	++	++	++	●	●
20	Formal Methods	D.28	○	+	+	++	++
21	Modelling	Table A.17	+✔	+✔	+✔	++✔	++✔
22	Structured Methodology	D.52	+	++	++	++	++
23	Modelling supported by computer aided design and specification tools	Table A.17	+	+	+	++	++

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.3 — Software architecture (7.3)

Table A.3 specifies about software architecture.

As an example section D.14 refers to techniques that can be used during programming to check control and data anomalies. TBvision helps to detect data flow anomalies such as UR (Undefined Referenced), DU (Defined Unreferenced) and DD (Defined and Defined) which are present in the developed software. Graphical callgraph displays help to highlight control flow anomalies such as uncalled procedures, unresolved functions, pointers and recursive functions. TBeXtreme® supports ranged testing for the different variable data types, where boundary value analysis can be performed.

Section D.19 refers to error detecting codes, where special values or combinations of values can be provided to check error-prone behavior. Black Box Testing using TBrun helps to verify the actual results with the expected results. All of the executed test cases in TBrun can be stored in TCF (Test Case File) format and then utilised to perform regression testing and checks for the presence of unexecuted paths in the software.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Formal methods	D.28	○	+	+	++	++
2	Modelling	Table A.17	+✔	++✔	++✔	++✔	++✔
3	Structured Methodology	D.52	+	++	++	++	++
4	Modular Approach	D.38	++✔	●✔	●✔	●✔	●✔
5	Components	Table A.20	++	++	++	++	++
6	Design and Coding Standards	Table A.12	++✔	++✔	++✔	●✔	●✔

7	Analysable Programs	D.2	++	++	++	++	++
8	Strongly Typed Programming Language	D.49	+	++	++	++	++
9	Structured Programming	D.53	+✔	++✔	++✔	++✔	++✔
10	Programming Language	Table A.15	+✔	++✔	++✔	++✔	++✔
11	Language Subset	A.35	○	○	○	++	++
12	Object Oriented Programming	Table A.22 D.57	+	+	+	+	+
13	Procedural Programming	D.60	+	++	++	++	++
14	Metaprogramming	D.59	+	+	+	+	+

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.4 — Software design and implementation (7.4)

Table A.4 refers to software design and implementation techniques (7.4) to develop software component design that achieves the requirements of the software design specification, to the extent required by the safety integrity level.

TBreq helps to generate traceability information to check the correctness of design by synchronising the requirement specification with the application code. Here, the application code design can be checked against industry defined metrics, which can be used to determine the complexity of developed software used in safety critical applications.

The LDRA tool suite enforces programming standards compliance and functional and structural coverage analysis for the C, C++, Ada and Java languages.

TBvision supports a wide range of industrial coding standards such as MISRA C:2012, MISRA C:2004, MISRA C: 1998, MISRA C++:2008, CERT, DERA C, JSF AV ++, etc. It also supports user configurable rules.

Section D.38 describes the modular approach, where the quality of the source code will be checked by decomposing the software into small comprehensible parts, in order to limit the complexity of the software. TBvision supports this activity by measuring the complexity of the source code against a required level of threshold values defined in design and coding standards.

Section D.53 is satisfied with TBvision which produces the Quality Review report in which the source code is checked against complexity thresholds, loop nesting depth, the number of procedures in a module, etc. and produces a pass/fail result.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Formal Proof	D.29	○	+	+	++	++
2	Static Analysis	Table A.19	○	++✔	++✔	++✔	++✔
3	Dynamic Analysis and Testing	Table A.13	○	++✔	++✔	++✔	++✔

4	Metrics	D.37	○	+✔	+✔	+✔	+✔
5	Traceability	D.58	+	++✔	++✔	●✔	●✔
6	Software Error Effect Analysis	D.25	○	+	+	++	++
7	Test Coverage for Code	Table A.21	+✔	++✔	++✔	++✔	++✔
8	Functional/ Black-box Testing	Table A.14	++✔	++✔	++✔	●✔	●✔
9	Performance Testing	Table A.18	○	++✔	++✔	++✔	++✔
10	Interface Testing	D.34	++✔	++✔	++✔	++✔	++✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.5 — Verification and testing (6.2 and 7.3)

Table A.5 refers to verification and testing. TBvision performs static analysis, where the source code is checked against programming guidelines such as MISRA, CERT, JSF++ AV, etc. to find different vulnerabilities and latent errors.

Section D.37 refers to different quality metrics. TBvision supports quality review of the source code and reports against the metrics listed below:

- Clarity
- Maintainability
- Testability
- Complexity Metrics
  - Includes knots, cyclomatic complexity, essential knots, essential cyclomatic complexity
- Hallstead's Metrics
  - Includes total operators, total operands, unique operators, unique operands vocabulary, length and volume
- Loop/Interval Analysis
  - Includes number of loops, depth of loop nesting, number of order 1 intervals, maximum interval nesting

Section D.58 refers to traceability between the requirements and the software development lifecycle. TBreq generates traceability reports for: systems requirements to software requirements, software requirements to software architecture, software architecture to software source code and software source code to test cases/ test results. TBmanager® supports the traceability between high level/low level requirements to the source code. The source code procedures can be mapped to each requirement and requirement verification can be performed.

Item 5, 7, 8, 9 and 10 can be satisfied with TBvision, TBrun and TBeXtreme, where functionality testing, performance testing and structural coverage analysis is performed on the source code. TBeXtreme supports automatic test cases generation with a single click, which generates multiple test cases with different inputs and supports the possible achievement of maximum coverage as determined by the source code structure. The remaining coverage can be supported by TBrun, by providing the inputs to the test cases required to achieve 100% structural coverage. The LDRA tool suite supports structural coverage of statement, branch/decision, MC/ DC (Modified Coverage/Decision Coverage), procedure/call coverage, etc.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Functional and Black-Box Testing	Table A.14	++✔	++✔	++✔	++✔	++✔
2	Performance Testing	Table A.18	○	+✔	+✔	++✔	++✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.6 – Integration (7.6)

Table A.6 refers to integration testing, where the developed software is tested on target hardware. TBrun and the TLP (Target Licensing Package) support integration testing on target hardware, where the input can be specified and the expected output can be checked to perform functional black-box testing. TBobjectBox® supports assembly source code coverage on different processors such as Intel, PowerPC, etc. TBrun supports timing analysis, which can be performed to check whether the function is executed within a given period of time.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Performance Testing	Table A.18	○	++✔	++✔	●✔	●✔
2	Functional and Black-Box Testing	Table A.14	++✔	++✔	++✔	●✔	●✔
3	Modelling	Table A.17	○	+✔	+✔	+✔	+✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.7 – Overall software testing (6.2 and 7.7)

Table A.7 refers to overall software testing, which covers performance testing and functional and black-box testing at the system level. The LDRA tool suite supports system level testing and produces a Test Manager report which is a consolidated report that keeps track of unit testing, module testing, integration testing and system testing providing a pass/fail result.

If a model based approach is used, the source code generated from the model should be verified against the requirements. The LDRA tool suite provides integration with different modeling tools such as IBM® Rational® Rhapsody®, MathWorks® Simulink® and Esterel® SCADE®, through which the source generated from these models is validated to perform functional/black-box testing and structural coverage analysis.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Static Software Analysis	D.13 D.37 Table A.19	+✔	++✔	++✔	++✔	++✔
2	Dynamic Software Analysis	Table A.13 Table A.14	○	+✔	+✔	++✔	++✔
3	Cause Consequence Diagrams	D.6	+	+	+	+	+
4	Event Tree Analysis	D.22	○	+	+	+	+
5	Software Error Effect Analysis	D.25	○	+✔	+✔	++✔	++✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.8 — Software analysis techniques (6.3)

Table A.8 refers to software analysis techniques, which covers software validation techniques. The LDRA tool suite supports static analysis and dynamic analysis to satisfying all techniques of software validation.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Accredited to EN ISO 9001	7.1	+	++	++	++	++
2	Compliant with EN ISO 9001	7.1	●	●	●	●	●
3	Compliant with ISO/IEC 90003	7.1	+	+	+	+	+
4	Company Quality System	7.1	●✔	●✔	●✔	●✔	●✔
5	Software Configuration Management	D.48	●	●	●	●	●
6	Checklists	D.7	+✔	++✔	++✔	++✔	++✔
7	Traceability	D.58	+✔	++✔	++✔	●✔	●✔
8	Data Recording and Analysis	D.12	++✔	++✔	++✔	●✔	●✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.9 — Software Quality Assurance (6.5)

Table A.9 refers to software quality assurance, which identifies, monitors and controls technical and managerial activities, which are necessary to ensure that the software achieves the required level of quality. TBmanager supports this activity in creating company specific quality assurance objectives and then keeps track of the status of the objectives - whether they are unfulfilled, partially fulfilled or fulfilled. It also documents and maintains desired assets and artifacts produced for each objective. During objective fulfillment, if any defects are raised, TBmanager keeps track of all defects and produces a defect report. These defect reports can then be closed within TBmanager once verified.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Impact Analysis	D.32	+✔	++✔	++✔	●✔	●✔
2	Data Recording and Analysis	D.12	++✔	++✔	++✔	●✔	●✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.10 —Software maintenance (9.2)

Table A.10 refers to software maintenance. Impact analysis aims to determine the extent to which a change or an enhancement to a software system has impacted upon the existing system. After the analysis has been completed, a decision is required concerning the verification of the software system. This depends on the number of software modules affected, the criticality of the affected software modules and the nature of the change. Possible decisions are:

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

TBreq produces impact analysis reports (upstream and downstream), which indicate how the software has changed when compared against an earlier version.

The TBrun regression analysis feature can be used to verify whether the newly introduced modules have affected the functionality of the existing system.

The complete system can be revalidated using the powerful static and dynamic analysis features of the LDRA tool suite.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Tabular Specification Methods	D.68	+	+	+	+	+
2	Application Specific Language	D.69	+	+	+	+	+
3	Simulation	D.42	+	++	++	++	++
4	Functional Testing	D.42	●✔	●✔	●✔	●✔	●✔
5	Checklists	D.7	+	++	++	●	●
6	Fagan Inspection	D.23	○	+	+	+	+
7	Formal Design Reviews	D.56	+	++	++	++	++
8	Formal Proof of Correctness (of data)	D.29	○	○	○	++	++
9	Walkthrough	D.56	+✔	+✔	+✔	++✔	++✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.11 — Data preparation techniques (8.4)

Table A.11 refers to data preparation techniques for software requirements, software architecture and development of software.

Section D.68 refers to tabular specification methods to provide a standardised and well structured means of defining data driven functions of the system.

TBmanager supports configuration of the objectives (checklists) and accordingly inputs and outputs can be defined, so that these objectives can be fulfilled once the programs are produced.

The LDRA tool suite automates Design Review, Code Review and Quality Review to detect errors in software during the development process.

A.2 Detailed Tables:

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Coding Standard	D.15	++✔	++✔	++✔	●✔	●✔
2	Coding Style guide	D.15	++✔	++✔	++✔	++✔	++✔
3	No Dynamic Objects	D.15	○	+✔	+✔	++✔	++✔
4	No Dynamic Variables	D.15	○	+✔	+✔	++✔	++✔
5	Limited use of Pointers	D.15	○	+✔	+✔	+✔	+✔
6	Limited use of Recursion	D.15	○	+✔	+✔	++✔	++✔
7	No Unconditional Jumps	D.15	○	++✔	++✔	++✔	++✔
8	Limited Size and Complexity of Functions, Subroutines and Methods	D.38	++✔	++✔	++✔	++✔	++✔
9	Entry/Exit Point Strategy for Functions, Subroutines and Methods	D.38	+✔	++✔	++✔	++✔	++✔
10	Limited Number of Subroutine Parameters	D.38	+✔	+✔	+✔	+✔	+✔
11	Limited use of Global Variables	D.38	++✔	++✔	++✔	●✔	●✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.12 — Coding standards

Table A.12 refers to design and coding standards. To reduce the likelihood of errors in the safety-related code and to facilitate its verification, use of an appropriate coding standard is a recommended practice in EN 50128. The LDRA tool suite supports many industrial programming standards, such as MISRA C:2012, MISRA C:2004, JSF AV++, CERT C, etc. and can be configured to support customer or domain specific requirements.

The application code can be verified with respect to the programming standards by the LDRA tool suite to develop an efficient system.

TBvision can be used to check the interrupt routines, pointers, recursion, and non structured control flow and run time checks. Structured Programming Verification (SPV) allows for the determination of the unstructured parts of the application code, which may lead to failures.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Test Case Execution from Boundary Value Analysis	D.4	○	++✔	++✔	++✔	++✔
2	Test Case Execution from Error Guessing	D.20	+✔	+✔	+✔	+✔	+✔
3	Test Case Execution from Error Seeding	D.21	○	+✔	+✔	+✔	+✔
4	Performance Modelling	D.39	○	+✔	+✔	++✔	++✔
5	Equivalence Classes and Input Partition Testing	D.18	○	+✔	+✔	++✔	++✔
6	Structure-Based Testing	D.50	○	+✔	+✔	++✔	++✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.13 – Dynamic analysis and testing

Table A.13 refers to dynamic analysis and testing referenced by clauses 11 and 14.

Item 1 mentions boundary value analysis which is used to detect software errors occurring at parameter limits or boundaries. TBrun can automatically generate test drivers and harnesses (wrapper code), with no extra coding or scripting required, enabling tests to be easily and efficiently run on code units. These tests can be automatically regressed with clear maintenance tracking and storage of test data and results. The test data maintenance process is streamlined through automatic detection of changes in source code and documentation of changes required in tests. TBrun can be used to perform boundary value analysis, functional testing and structural coverage analysis.

Structure based testing can be performed using the LDRA tool suite to determine the required coverage information. Coverage metrics linked to the execution of statements, branch/decisions, compound conditions (MC/DC), LCSAJ and basis path can all be checked as required by the standard.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Test Case Execution from Cause Consequence Diagrams	D.6	○	○	○	+✔	++✔
2	Prototyping/Animation	D.43	○	○	○	+	+
3	Boundary Value Analysis	D.4	+✔	++✔	++✔	++✔	++✔
4	Equivalence Classes and Input Partition Testing	D.18	+✔	++✔	++✔	++✔	++✔
5	Process Simulation	D.42	+	+	+	++	++

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.14 – Functional/black-box testing

Table A.14 refers to functional/black box test referenced by clauses 10, 12, 13 and 14. The LDRA tool suite provides integration with modeling tools such as IBM® Rational® Rhapsody®, through which it can create test cases from sequence diagrams, classes and packages. TBrun can be used to perform boundary value analysis. Through an integration with MathWorks® Simulink®, test cases are generated based on simulation test cases and can perform complete structural coverage for the model generated source code. The LDRA tool suite also integrates with Esterel® SCADE®, to generate Test Code Files (TCF)s from the model and these TCFs can then be imported into TBrun to execute the test cases.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	ADA	D.54	+✔	++✔	++✔	++✔	++✔
2	MODULA-2	D.54	+	++	++	++	++
3	PASCAL	D.54	+	++	++	+	+
4	C or C++	D.54 D.35	+✔	+✔	+✔	+✔	+✔
5	PL/M	D.54	+	+	+	○	○
6	BASIC	D.54	+	○	○	○	○
7	Assembler	D.54	+✔	+✔	+✔	+✔	+✔
8	C#	D.54	+	+	+	+	+
9	JAVA	D.54	+✔	+✔	+✔	+✔	+✔
10	Statement List	D.54	+	+	+	+	+

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.15 — Textual programming languages

Table A.15 refers to programming languages (D.4) referenced by clause 10. The LDRA tool suite supports C, C++, Ada and Java as high-level programming languages. In addition the LDRA tool suite also supports assembly languages for processors such as Intel-x86 family, Freescale-PowerPC, Texas Instruments-Hercules, etc.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Functional Block Diagrams	D.63	+✔	+✔	+✔	+✔	+✔
2	Sequential Function Charts	D.61	○	++✔	++✔	++✔	++✔
3	Ladder Diagrams	D.62	+	+	+	++	++
4	State Charts	D.64	+	++	++	++	++

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.16 — Diagrammatic languages for application algorithms

Table (A.16) refers on the usage of Diagrammatic Languages for Application Algorithms.

Section D.63 recommends diagrammatic representation of a function between input variables and output variables. Section D.62 describes a program in a diagrammatic way. The callgraphs and the flowgraphs generated by the LDRA tool suite can be used to represent the control flow and data flow information graphically or diagrammatically.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Data Modelling	D.65	+	+	+	++	++
2	Data Flow Diagrams	D.11	○	+	+	++	++
3	Control Flow Diagrams	D.66	+	+	+	++	++
4	Finite State Machines or State Transition Diagrams	D.27	○	++	++	++	++
5	Time Petri Nets	D.55	○	+	+	++	++
6	Decision/Truth Tables	D.13	+	+	+	++	++
7	Formal Methods	D.28	○	+	+	++	++
8	Performance Modelling	D.39	○	+	+	++	++
9	Prototyping/Animation	D.43	○	+	+	+	+
10	Structure Diagrams	D.51	○	+	+	++	++
11	Sequence Diagrams	D.67	+	++	++	++	++

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.17 — Modelling

Table A.17 refers to Modelling (D.5) referenced by Clause 13. The LDRA tool suite provides integration with different modelling tools such as IBM® Rational® Rhapsody®, MathWorks® Simulink®, Esterel® SCADE® to perform code review, and unit tests for source code generated from these modelling tools.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Avalanche/Stress Testing	D.3	○	+	+	++	++
2	Response Timing and Memory Constraints	D.45	○	++	++	++	++
3	Performance Requirements	D.40	○	++✔	++✔	++✔	++✔

Requirement
A suitable set of techniques shall be chosen according to the software safety integrity level.

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.18 – Performance testing

Table A.18 refers to performance testing. TBrun supports performance testing on target, where timing analysis can be performed to check whether a function is executed in a given period of time.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Boundary Value Analysis	D.4	○	+✔	+✔	++✔	++✔
2	Checklists	D.7	○	+	+	+	+
3	Control Flow Analysis	D.8	○	++✔	++✔	++✔	++✔
4	Data Flow Analysis	D.10	○	++✔	++✔	++✔	++✔
5	Error Guessing	D.20	○	+✔	+✔	+✔	+✔
6	Walkthroughs/Design Reviews	D.56	++	++	++	++	++

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.19 — Static analysis (D.8)

Table A.19 refers to static analysis (D.8) referenced by clauses 11 and 14. Static analysis helps to detect code that may lead to run time errors and general programming errors as per the programming guidelines. This can be addressed by TBvision static checks for behaviors such as:

- Divide by Zero
- Out of Bounds Array Index
- Uninitialised Local Variable
- Illegal Dereference of Pointer
- Uninitialised Pointer
- Uninitialised Variable
- Non Termination of Loop
- Non Termination of Call

TBvision supports a wide range of programming standards for the C, C++ and Ada and Java languages. It also supports the addition and application of user-defined standards.

Item 3 and Item 4 are satisfied with Static and Dynamic Callgraphs and Flowgraphs generated by TBvision and TBrun. These represent the function blocks executed in the software as well as the control flow paths. They also represent the data flow information. Data flow diagrams document how data input is transformed to output, within each stage in the diagram, representing a distinct transformation. Annotated source code provides information on how the data is transferred between each module. Dynamic data flow coverage reports the coverage details of data at runtime.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Information Hiding	D.33	○	○	○	○	○
2	Information Encapsulation	D.33	+✔	++✔	++✔	++✔	++✔
3	Parameter Number Limit	D.38	+✔	+✔	+✔	+✔	+✔
4	Fully Defined Interface	D.38	+✔	++✔	++✔	●✔	●✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.20 — Components

Table A.20 refers to components. TBvision supports quality review configurations, where quality thresholds such as cyclomatic complexity, parameter numbers, etc. can be configured, and the software can then be checked against these threshold values.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Statement	D.50	+✔	++✔	++✔	++✔	++✔
2	Branch	D.50	○	+✔	+✔	++✔	++✔
3	Compound Condition	D.50	○	+✔	+✔	++✔	++✔
4	Data flow	D.50	○	+✔	+✔	++✔	++✔
5	Path	D.50	○	+✔	+✔	++✔	++✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.21 — Test Coverage for Code

Table A.21 refers to different metrics for structural coverage of source code.

Section D.50 mentions Structure-based testing, i.e. "Based on analysis of the program, a set of input data is chosen so that a large (and often pre-specified target) percentage of the program code is exercised. Measures of code coverage will vary as follows, depending upon the level of rigour required. In all cases, a high percentage of the selected coverage metrics (ideally 100%) should be the aim. If it is not possible to achieve 100% coverage, the reasons why 100% cannot be achieved should be documented in the test report (for example, defensive code which can only be entered if a hardware problem arises)."

The coverage metrics are:

- Statement Coverage
- Branch Coverage
- Compound Condition (MC/DC) Coverage
- Data Flow Coverage
- Path Coverage

LDRA Dynamic Coverage Analysis helps to achieve the required level of coverage for EN 50128 compliance. In addition, a Tool Qualification Support Pack is available to certify the LDRA tool suite for the current project, up to the required SIL Levels.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Traceability of the Concept of the Application Domain to the Classes of the Architecture	-	+✔	+✔	+✔	+✔	+✔
2	Use of Suitable Frames, Commonly Used Combinations of Classes and Design Patterns	-	+✔	+✔	+✔	++✔	++✔
3	Object Oriented Detailed Design	Table A.23	+✔	+✔	+✔	++✔	++✔

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.22 — Object oriented software architecture

Table A.22 refers to object oriented software architecture. TBreq supports the traceability from system level requirements to software requirements and software requirements to software architecture to fulfill item 1 of the above table. The LDRA tool suite supports verification of object oriented programming languages such as C++, where traceability can be achieved throughout the software development lifecycle with static analysis and dynamic analysis used to perform functional testing and structural coverage testing.

	TECHNIQUE/MEASURE	REF.	SWSIL				
			0	1	2	3	4
1	Classes Should Have Only One Objective	-	+✔	+✔	+✔	++✔	++✔
2	Inheritance Used Only if the Derived Class is a Refinement of its Basic Class	-	+✔	++✔	++✔	++✔	++✔
3	Depth of Inheritance Limited by Coding Standards	-	+✔	+✔	+✔	++✔	++✔
4	Overriding of Operations (Methods) Under Strict Control	-	+✔	+✔	+✔	++✔	++✔
5	Multiple Inheritance Used Only For Interface Classes	-	+✔	++✔	++✔	++✔	++✔
6	Inheritance From Unknown Classes	-	○	○	○	○	○

"●" The method is mandatory for this SIL.
"++" The method is highly recommended for this SIL.
"+" The method is recommended for this SIL.
"○" The method has no recommendation for or against its usage for this SIL.
"✔" Satisfied by the LDRA tool suite.

Table A.23 – Object oriented detailed design

Table A.23 refers to object oriented detailed design. TBvision quality review identifies and reports the depth of inheritance, base classes and derived classes. The threshold values can be configured as per coding standards and TBvision detects and reports the violation, if the software value exceeds the threshold value.

Conclusion

It is clear from this document that the LDRA tool suite can contribute extensively to the achievement of certification to the EN 50128 standard. It provides suitable facilities for any SIL. The tool maturity and longevity are second to none.

20 years ago, LDRA completed ISO 9000 compliance adding ISO 9001:2008 compliance five years later. For more than 40 years, LDRA has been directly involved in various safety standards, actively chairing standards committees and providing technical expertise in working committees.

The LDRA tool suite is a mature product which is in widespread use all around the world in the rail, aerospace and automotive domains and has been used to facilitate software certification to many international standards including EN 50128.

Availability

For availability information regarding the LDRA tool suite, please refer to LDRA website, or contact LDRA.

Proven Track Record in Meeting Industry Standards

LDRA has an established track record in the area of programming standards enforcement, as well as being a key contributor to the MISRA C and C++ standard. LDRA has played an active role on the MISRA C++ committee in providing both committee members and the chairman from the ranks of LDRA employees. LDRA is also represented on the MISRA C committee with three members of the LDRA technical team. It is no surprise, therefore, that the LDRA tool suite provides the most comprehensive C and C++ coding standards enforcement available in the market today.

Today the LDRA tool suite is being utilised by companies around the world to numerous standards, including EN 50128 and has been used in safety-critical projects for over twenty years. For a cross section of LDRA customers, please refer to the Client List available on our website.

Flexible Tool Support

As certification authorities gain further experience in applying EN 50128 to projects, customers must be concerned that their tool vendors have the flexibility to cope with changing requirements. LDRA is committed to helping existing customers meet changing requirements now and in the future.

Ease of Use

The tool's ease of use is a key issue when establishing project procedures. The LDRA tool suite has been specifically developed to enable simple measurement of conformance to the various classes of EN 50128. Reports are tailored to give users EN 50128 information quickly and concisely, speeding up the testing procedure. Reports can be produced in either ASCII or HTML formats. Either format can be easily incorporated into a word processor or DTP system. HTML has the added advantage of links and the ability to be published on an intranet. Facilities are being developed to allow customers to generate custom reports directly from the analysis data.

LDRA Client References

LDRA's products and services are widely used by companies whose names are synonymous with embedded railway applications including GE Transportation, BTR rail, Bombardier Transport, Nippon Signal, ADtranz, Westinghouse.

Certificate Number FM 26376

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

LDRA Technology Pvt. Ltd.
#2989/1B, 3rd Floor, 12th Main, 80 Feet Road,
HAL II Stage, Bangalore- 560008. Near BSNL Building
Tel: +91 80 4080 8707
e-mail: india@ldra.com