



Institut luxembourgeois de la normalisation
de l'accréditation, de la sécurité et qualité
des produits et services

ILNAS-EN 50128:2011

Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems

Bahnanwendungen -
Telekommunikationstechnik,
Signaltechnik und
Datenverarbeitungssysteme - Software
Applications ferroviaires - Systèmes de
signalisation, de télécommunication et
de traitement - Logiciels pour systèmes
de commande et de protection

06/2011



National Foreword

This European Standard EN 50128:2011 was adopted as Luxembourgish Standard ILNAS-EN 50128:2011.

Every interested party, which is member of an organization based in Luxembourg, can participate for FREE in the development of Luxembourgish (ILNAS), European (CEN, CENELEC) and International (ISO, IEC) standards:

- Participate in the design of standards
- Foresee future developments
- Participate in technical committee meetings

<https://portail-qualite.public.lu/fr/normes-normalisation/participer-normalisation.html>

THIS PUBLICATION IS COPYRIGHT PROTECTED

Nothing from this publication may be reproduced or utilized in any form or by any mean - electronic, mechanical, photocopying or any other data carries without prior permission!

EUROPEAN STANDARD
NORME EUROPÉENNE
EUROPÄISCHE NORM

EN 50128

June 2011

ICS 35.240.60; 45.020; 93.100

Supersedes EN 50128:2001

English version

**Railway applications -
Communication, signalling and processing systems -
Software for railway control and protection systems**

Applications ferroviaires -
Systèmes de signalisation, de
télécommunication et de traitement -
Logiciels pour systèmes de commande et
de protection ferroviaire

Bahnanwendungen -
Telekommunikationstechnik,
Signaltechnik und
Datenverarbeitungssysteme -
Software für Eisenbahnsteuerungs- und
Überwachungssysteme

This European Standard was approved by CENELEC on 2011-04-25. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Management Centre: Avenue Marnix 17, B - 1000 Brussels

Contents

Foreword	6
Introduction	7
1 Scope	10
2 Normative references	11
3 Terms, definitions and abbreviations	11
3.1 Terms and definitions.....	11
3.2 Abbreviations	15
4 Objectives, conformance and software safety integrity levels	16
5 Software management and organisation.....	17
5.1 Organisation, roles and responsibilities	17
5.2 Personnel competence.....	20
5.3 Lifecycle issues and documentation	21
6 Software assurance	23
6.1 Software testing	23
6.2 Software verification.....	25
6.3 Software validation	27
6.4 Software assessment	28
6.5 Software quality assurance.....	30
6.6 Modification and change control.....	33
6.7 Support tools and languages	34
7 Generic software development.....	37
7.1 Lifecycle and documentation for generic software	37
7.2 Software requirements	37
7.3 Architecture and Design.....	40
7.4 Component design	46
7.5 Component implementation and testing	49
7.6 Integration.....	50
7.7 Overall Software Testing / Final Validation	52
8 Development of application data or algorithms: systems configured by application data or algorithms.....	54

8.1 Objectives	54
8.2 Input documents	55
8.3 Output documents	55
8.4 Requirements	55
9 Software deployment and maintenance	60
9.1 Software deployment.....	60
9.2 Software maintenance.....	62
Annex A (normative) Criteria for the Selection of Techniques and Measures.....	65
A.1 Clauses tables	66
A.2 Detailed tables	73
Annex B (normative) Key software roles and responsibilities	79
Annex C (informative) Documents Control Summary	88
Annex D (informative) Bibliography of techniques.....	90
D.1 Artificial Intelligence Fault Correction.....	90
D.2 Analysable Programs	90
D.3 Avalanche/Stress Testing	91
D.4 Boundary Value Analysis	91
D.5 Backward Recovery	92
D.6 Cause Consequence Diagrams.....	92
D.7 Checklists	92
D.8 Control Flow Analysis.....	93
D.9 Common Cause Failure Analysis	93
D.10 Data Flow Analysis.....	94
D.11 Data Flow Diagrams	94
D.12 Data Recording and Analysis.....	95
D.13 Decision Tables (Truth Tables).....	95
D.14 Defensive Programming	96
D.15 Coding Standards and Style Guide.....	96
D.16 Diverse Programming	97
D.17 Dynamic Reconfiguration.....	98
D.18 Equivalence Classes and Input Partition Testing.....	98
D.19 Error Detecting and Correcting Codes.....	98
D.20 Error Guessing.....	99
D.21 Error Seeding.....	99
D.22 Event Tree Analysis	99
D.23 Fagan Inspections.....	100
D.24 Failure Assertion Programming	100
D.25 SEEA – Software Error Effect Analysis.....	100
D.26 Fault Detection and Diagnosis	101
D.27 Finite State Machines/State Transition Diagrams.....	102
D.28 Formal Methods	102
D.29 Formal Proof	108

D.30	Forward Recovery.....	108
D.31	Graceful Degradation.....	108
D.32	Impact Analysis.....	109
D.33	Information Hiding / Encapsulation	109
D.34	Interface Testing	110
D.35	Language Subset.....	110
D.36	Memorising Executed Cases	110
D.37	Metrics	111
D.38	Modular Approach.....	111
D.39	Performance Modelling	112
D.40	Performance Requirements.....	112
D.41	Probabilistic Testing.....	113
D.42	Process Simulation	113
D.43	Prototyping / Animation	114
D.44	Recovery Block	114
D.45	Response Timing and Memory Constraints.....	114
D.46	Re-Try Fault Recovery Mechanisms.....	115
D.47	Safety Bag	115
D.48	Software Configuration Management	115
D.49	Strongly Typed Programming Languages	115
D.50	Structure Based Testing	116
D.51	Structure Diagrams.....	116
D.52	Structured Methodology.....	117
D.53	Structured Programming.....	117
D.54	Suitable Programming languages.....	118
D.55	Time Petri Nets	119
D.56	Walkthroughs / Design Reviews	119
D.57	Object Oriented Programming	119
D.58	Traceability.....	120
D.59	Metaprogramming.....	121
D.60	Procedural programming	121
D.61	Sequential Function Charts.....	121
D.62	Ladder Diagram	122
D.63	Functional Block Diagram	122
D.64	State Chart or State Diagram	122
D.65	Data modelling	122
D.66	Control Flow Diagram/Control Flow Graph	123
D.67	Sequence diagram.....	124
D.68	Tabular Specification Methods	124
D.69	Application specific language.....	124
D.70	UML (Unified Modeling Language)	125
D.71	Domain specific languages.....	126

Bibliography	127
---------------------------	------------

Figures

Figure 1 – Illustrative Software Route Map	9
Figure 2 – Illustration of the preferred organisational structure	18
Figure 3 – Illustrative Development Lifecycle 1	22
Figure 4 – Illustrative Development Lifecycle 2	23

Tables

Table 1 - Relation between tool class and applicable sub-clauses	37
Table A.1– Lifecycle Issues and Documentation (5.3)	66
Table A.2 – Software Requirements Specification (7.2).....	68
Table A.3 – Software Architecture (7.3).....	69
Table A.4– Software Design and Implementation (7.4).....	70
Table A.5 – Verification and Testing (6.2 and 7.3)	71
Table A.6 – Integration (7.6).....	71
Table A.7 – Overall Software Testing (6.2 and 7.7).....	71
Table A.8 – Software Analysis Techniques (6.3).....	72
Table A.9 – Software Quality Assurance (6.5).....	72
Table A.10 – Software Maintenance (9.2)	72
Table A.11 – Data Preparation Techniques (8.4)	73
Table A.12 – Coding Standards.....	73
Table A.13 – Dynamic Analysis and Testing	74
Table A.14 – Functional/Black Box Test.....	74
Table A.15 – Textual Programming Languages	75
Table A.16 – Diagrammatic Languages for Application Algorithms	75
Table A.17 – Modelling	76
Table A.18 – Performance Testing	76
Table A.19 – Static Analysis	76
Table A.20 – Components	77
Table A.21 – Test Coverage for Code	77
Table A.22 – Object Oriented Software Architecture.....	78
Table A.23 – Object Oriented Detailed Design.....	78
Table B.1 – Requirements Manager Role Specification	79
Table B.2 – Designer Role Specification	80
Table B.3 – Implementer Role Specification.....	81
Table B.4 – Tester Role Specification	82
Table B.5 – Verifier Role Specification	83
Table B.6 – Integrator Role Specification	84
Table B.7 – Validator Role Specification.....	85
Table B.8 – Assessor Role Specification.....	86
Table B.9 – Project Manager Role Specification	87
Table B.10 – Configuration Manager Role Specification	87
Table C.1 – Documents Control Summary.....	88

Foreword

This European Standard was prepared by SC 9XA, Communication, signalling and processing systems, of Technical Committee CENELEC TC 9X, Electrical and electronic applications for railways. .

It was submitted to the Formal Vote and was approved by CENELEC as EN 50128 on 2011-04-25.

This document supersedes EN 50128:2001.

The main changes with respect to EN 50128:2001 are listed below:

- requirements on software management and organisation, definition of roles and competencies, deployment and maintenance have been added;
- a new clause on tools has been inserted, based on EN 61508-2:2010;
- tables in Annex A have been updated.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN and CENELEC shall not be held responsible for identifying any or all such patent rights.

The following dates were fixed:

- | | | |
|--|-------|------------|
| – latest date by which the EN has to be implemented at national level by publication of an identical national standard or by endorsement | (dop) | 2012-04-25 |
| – latest date by which the national standards conflicting with the EN have to be withdrawn | (dow) | 2014-04-25 |

This European Standard should be read in conjunction with EN 50126-1:1999 "*Railway applications – The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) – Part 1: Basic requirements and generic process*" and EN 50129:2003 "*Railway applications – Communication, signalling and processing systems – Safety related electronic systems for signalling*".

Introduction

This European Standard is part of a group of related standards. The others are EN 50126-1:1999 "*Railway applications – The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) – Part 1: Basic requirements and generic process*" and EN 50129:2003 "*Railway applications – Communication, signalling and processing systems – Safety related electronic systems for signalling*".

EN 50126-1 addresses system issues on the widest scale, while EN 50129 addresses the approval process for individual systems which can exist within the overall railway control and protection system. This European Standard concentrates on the methods which need to be used in order to provide software which meets the demands for safety integrity which are placed upon it by these wider considerations.

This European Standard provides a set of requirements with which the development, deployment and maintenance of any safety-related software intended for railway control and protection applications shall comply. It defines requirements concerning organisational structure, the relationship between organisations and division of responsibility involved in the development, deployment and maintenance activities. Criteria for the qualification and expertise of personnel are also provided in this European Standard.

The key concept of this European Standard is that of levels of software safety integrity. This European Standard addresses five software safety integrity levels where 0 is the lowest and 4 the highest one. The higher the risk resulting from software failure, the higher the software safety integrity level will be.

This European Standard has identified techniques and measures for the five levels of software safety integrity. The required techniques and measures for software safety integrity levels 0-4 are shown in the normative tables of Annex A. In this version, the required techniques for level 1 are the same as for level 2, and the required techniques for level 3 are the same as for level 4. This European Standard does not give guidance on which level of software safety integrity is appropriate for a given risk. This decision will depend upon many factors including the nature of the application, the extent to which other systems carry out safety functions and social and economic factors.

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

This European Standard specifies those measures necessary to achieve these requirements.

EN 50126-1 and EN 50129 require that a systematic approach be taken to

- a) identify hazards, assessing risks and arriving at decisions based on risk criteria,
- b) identify the necessary risk reduction to meet the risk acceptance criteria,
- c) define an overall System Safety Requirements Specification for the safeguards necessary to achieve the required risk reduction,
- d) select a suitable system architecture,
- e) plan, monitor and control the technical and managerial activities necessary to translate the System Safety Requirements Specification into a Safety-Related System of a validated safety integrity.

As decomposition of the specification into a design comprising safety-related systems and components takes place, further allocation of safety integrity levels is performed. Ultimately this leads to the required software safety integrity levels.

The current state-of-the-art is such that neither the application of quality assurance methods (so-called fault avoiding measures and fault detecting measures) nor the application of software fault tolerant approaches can guarantee the absolute safety of the software. There is no known way to prove the absence of faults in reasonably complex safety-related software, especially the absence of specification and design faults.