



**SLOVENSKI STANDARD**  
**oSIST prEN 50128:2009**

**01-oktober-2009**

---

Željezniške aplikacije - Komunikacija, signalizacija in obratniški sistemi - Programi za nadzor in zaščito železniških aplikacij

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

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

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

**Ta slovenski standard je istoveten z: prEN 50128:2009**

<https://standards.iteh.ai/catalog/standards/sist/407b069f-cc90-4214-917b-d82b1ad241e7/sist-en-50128-2011>

**ICS:**

35.240.60	Uporabniške rešitve IT v transportu in trgovini	IT applications in transport and trade
45.020	Železniška tehnika na splošno	Railway engineering in general

**oSIST prEN 50128:2009**

**en,fr,de**



EUROPEAN STANDARD  
NORME EUROPÉENNE  
EUROPÄISCHE NORM

**DRAFT**  
**prEN 50128**

July 2009

ICS 35.240.60; 45.020; 93.100

Will supersede 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 draft European Standard is submitted to CENELEC members for CENELEC enquiry.  
Deadline for CENELEC: 2010-01-08.

It has been drawn up by CLC/SC 9XA.

If this draft becomes a European Standard, 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.

This draft European Standard was established by CENELEC 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, 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.

Warning : This document is not a European Standard. It is distributed for review and comments. It is subject to change without notice and shall not be referred to as a European Standard.

## CENELEC

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

**Central Secretariat: Avenue Marnix 17, B - 1000 Brussels**

1

## Foreword

2 This draft European Standard was prepared by SC 9XA, Communication, signalling and processing systems,  
3 of Technical Committee CENELEC TC 9X, Electrical and electronic applications for railways. It is submitted  
4 to the CENELEC enquiry.

5 This document will supersede EN 50128:2001.

6 The main changes with respect to the previous edition are listed below:

- 7 • requirements on software management and organisation, definition of roles and competencies,  
8 deployment and maintenance have been added;
- 9 • a new clause on tools has been inserted, based on EN 61508-2:2008;
- 10 • tables in Annex A have been updated.

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

15 This draft European Standard has been prepared under a mandate given to CENELEC by the European  
16 Commission and the European Free Trade Association and covers essential requirements of EC Directives  
17 2001/16/EC and 96/48/EC.

18

iteh Standards  
(<https://standards.itech.ai>)  
Document Preview

SIST EN 50128:2011

<https://standards.itech.ai/catalog/standards/sist/407b069f-cc90-4214-917b-d82b1ad241e7/sist-en-50128-2011>

19

**Contents**

20	<b>Introduction</b> .....	<b>7</b>
21	<b>1 Scope</b> .....	<b>10</b>
22	<b>2 Normative references</b> .....	<b>11</b>
23	<b>3 Terms, definitions and abbreviations</b> .....	<b>11</b>
24	3.1 Terms and definitions .....	11
25	3.2 Abbreviations .....	15
26	<b>4 Objectives, Conformance and Software Safety Integrity Levels</b> .....	<b>16</b>
27	<b>5 Software management and organisation</b> .....	<b>17</b>
28	5.1 Organisation, Roles and Responsibilities .....	17
29	5.2 Personnel Competence .....	19
30	5.3 Lifecycle issues and documentation .....	20
31	<b>6 Software assurance</b> .....	<b>22</b>
32	6.1 Software testing .....	22
33	6.2 Software verification.....	23
34	6.3 Software validation.....	25
35	6.4 Software assessment .....	26
36	6.5 Software quality assurance.....	28
37	6.6 Modification and change control .....	30
38	6.7 Support tools and languages .....	31
39	<b>7 Generic software development</b> .....	<b>34</b>
40	7.1 Lifecycle and documentation for generic software.....	34
41	7.2 Software requirements.....	35
42	7.3 Architecture and Design.....	37
43	7.4 Component design.....	43
44	7.5 Component implementation and testing .....	45
45	7.6 Integration .....	46
46	7.7 Final Acceptance/Validation.....	48
47	<b>8 Development of application data or algorithms: systems configured by application data or algorithms</b> .....	<b>50</b>
49	8.1 Objectives .....	50
50	8.2 Input documents .....	51
51	8.3 Output documents .....	51
52	8.4 Requirements .....	51
53	<b>9 Software deployment and maintenance</b> .....	<b>54</b>
54	9.1 Software deployment .....	54
55	9.2 Software maintenance .....	55
56	<b>Annex A (normative) Criteria for the Selection of Techniques and Measures</b> .....	<b>58</b>
57	A.1 Clauses tables .....	59
58	A.2 Detailed tables .....	65

59	<b>Annex B (normative) Key software roles and responsibilities .....</b>	<b>71</b>
60	<b>Annex C (informative) Documents Control Summary and Document Flow Diagrams .....</b>	<b>80</b>
61	<b>Annex D (informative) Bibliography of techniques.....</b>	<b>82</b>
62	D.1 AI Fault Correction .....	82
63	D.2 Analysable Programs .....	82
64	D.3 Avalanche/Stress Testing .....	83
65	D.4 Boundary Value Analysis .....	83
66	D.5 Backward Recovery .....	84
67	D.6 Cause Consequence Diagrams .....	84
68	D.7 Certified Tools and Certified Translators .....	84
69	D.8 Checklists .....	85
70	D.9 Control Flow Analysis.....	85
71	D.10 Common Cause Failure Analysis .....	86
72	D.11 Data Flow Analysis.....	86
73	D.12 Data Flow Diagrams .....	87
74	D.13 Data Recording and Analysis.....	87
75	D.14 Decision Tables (Truth Tables).....	88
76	D.15 Defensive Programming .....	88
77	D.16 Coding Standards and Style Guide.....	89
78	D.17 Diverse Programming .....	89
79	D.18 Dynamic Reconfiguration.....	90
80	D.19 Equivalence Classes and Input Partition Testing.....	90
81	D.20 Error Detecting and Correcting Codes.....	91
82	D.21 Error Guessing.....	91
83	D.22 Error Seeding.....	91
84	D.23 Event Tree Analysis .....	92
85	D.24 Fagan Inspections.....	92
86	D.25 Failure Assertion Programming .....	92
87	D.26 SEEA – Software Error Effect Analysis.....	93
88	D.27 Fault Detection and Diagnosis .....	94
89	D.28 Fault Tree Analysis .....	94
90	D.29 Finite State Machines/State Transition Diagrams.....	95
91	D.30 Formal Methods .....	95
92	D.31 Formal Proof .....	100
93	D.32 Forward Recovery.....	101
94	D.33 Graceful Degradation.....	101
95	D.34 Hazard and Operability Study (HAZOP) .....	101
96	D.35 Impact Analysis.....	102
97	D.36 Information Hiding / Encapsulation .....	103
98	D.37 Interface Testing .....	103
99	D.38 Language Subset.....	104
100	D.39 Memorising Executed Cases .....	104
101	D.40 Library of Trusted/Verified Components .....	104
102	D.41 Markov Models.....	105
103	D.42 Metrics .....	105

104	D.43 Modular Approach.....	106
105	D.44 Monte-Carlo Simulation .....	106
106	D.45 Performance Modelling.....	107
107	D.46 Performance Requirements.....	107
108	D.47 Probabilistic Testing.....	108
109	D.48 Process Simulation .....	108
110	D.49 Prototyping / Animation.....	109
111	D.50 Recovery Block.....	109
112	D.51 Reliability Block Diagram .....	109
113	D.52 Response Timing and Memory Constraints.....	110
114	D.53 Re-Try Fault Recovery Mechanisms.....	110
115	D.54 Safety Bag .....	110
116	D.55 Sneak Circuit Analysis .....	110
117	D.56 Software Configuration Management .....	111
118	D.57 Strongly Typed Programming Languages .....	111
119	D.58 Structure Based Testing .....	112
120	D.59 Structure Diagrams.....	112
121	D.60 Structured Methodology.....	113
122	D.61 Structured Programming.....	116
123	D.62 Suitable Programming languages.....	116
124	D.63 Symbolic Execution.....	117
125	D.64 Time Petri Nets.....	117
126	D.65 Translator Proven In Use.....	118
127	D.66 Walkthroughs / Design Reviews.....	118
128	D.67 Object Oriented Programming.....	119
129	D.68 Traceability.....	122
130	D.69 Static verification of runtime properties by abstract interpretation .....	122
131	D.70 Dataflow.....	123
132	D.71 Metaprogramming.....	123
133	D.72 Procedural programming .....	123
134	D.73 Sequential Function Charts.....	124
135	D.74 Ladder Diagram .....	124
136	D.75 Functional Block Diagram.....	124
137	D.76 State Chart or State Diagram .....	124
138	D.77 Data modelling.....	124
139	D.78 Control Flow Diagram/Control Flow Graph.....	125
140	D.79 Sequence diagram.....	126
141	<b>Bibliography.....</b>	<b>127</b>
142		

143 **Figures**

144	Figure 1 – Illustrative Software Route Map.....	9
145	Figure 2 – Independence and combination of roles versus software safety integrity levels.....	18
146	Figure 3 – Illustrative Development Lifecycle 1.....	21
147	Figure 4 – Illustrative Development Lifecycle 2.....	22

148 **Tables**

149	Table 1 – Relation between tool class and applicable subclauses.....	34
150	Table A.1 – Lifecycle Issues and Documentation (5.3).....	59
151	Table A.2 – Software Requirements Specification (7.2).....	60
152	Table A.3 – Software Architecture (7.3).....	61
153	Table A.4 – Software Design and Implementation (7.4).....	62
154	Table A.5 – Verification and Testing (6.2 and 7.3).....	63
155	Table A.6 – Integration (7.6).....	63
156	Table A.7 – Overall Software Testing (6.2 and 7.7).....	63
157	Table A.8 – Software Analysis Techniques (6.3).....	64
158	Table A.9 – Software Quality Assurance (6.5).....	64
159	Table A.10 – Software Maintenance (9.2).....	64
160	Table A.11 – Coding Standards.....	65
161	Table A.12 – Dynamic Analysis and Testing.....	65
162	Table A.13 – Functional/Black Box Test.....	66
163	Table A.14 – Textual Programming Languages.....	66
164	Table A.15 – Diagrammatic Languages for Application Algorithms.....	67
165	Table A.16 – Modelling.....	67
166	Table A.17 – Performance Testing.....	67
167	Table A.18 – Static Analysis.....	68
168	Table A.19 – Components.....	68
169	Table A.20 – Test Coverage for Code.....	69
170	Table A.21 – Object Oriented Software Architecture.....	70
171	Table A.22 – Object Oriented Detailed Design.....	70
172	Table B.1 – Requirements Manager Role Specification.....	71
173	Table B.2 – Designer Role Specification.....	72
174	Table B.3 – Implementer Role Specification.....	73
175	Table B.4 – Tester Role Specification.....	74
176	Table B.5 – Verifier Role Specification.....	75
177	Table B.6 – Integrator Role Specification.....	76
178	Table B.7 – Validator Role Specification.....	77
179	Table B.8 – Assessor Role Specification.....	78
180	Table B.9 – Project Manager Role Specification.....	79
181	Table B.10 – Configuration Manager Role Specification.....	79
182	Table C.1 – Documents Control Summary.....	80

183



## 184 Introduction

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

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

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

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

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

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

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

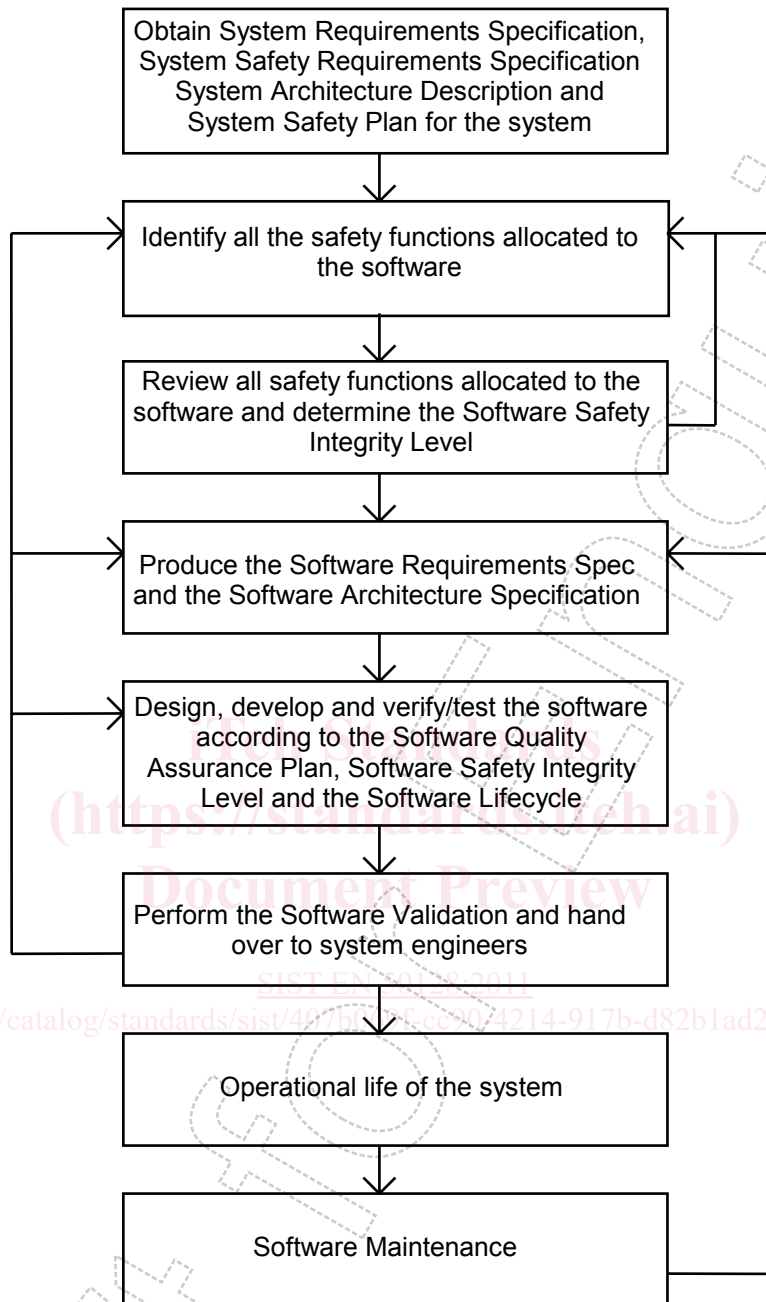
211 EN 50126-1 and EN 50129 require that a systematic approach be taken to

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

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

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

- 226 The principles applied in developing high integrity software include, but are not restricted to
- 227 – top-down design methods,
  - 228 – modularity,
  - 229 – verification of each phase of the development lifecycle,
  - 230 – verified components and component libraries;
  - 231 – clear documentation and traceability,
  - 232 – auditable documents,
  - 233 – validation,
  - 234 – assessment,
  - 235 – configuration management and change control and
  - 236 – appropriate consideration of organisation and personnel competency issues.
- 237 These and related principles must be correctly applied. This European Standard specifies the level of  
238 assurance required to demonstrate this at each software safety integrity level.
- 239 The System Safety Requirements Specification identifies all safety functions allocated to software and  
240 determines their system safety integrity level. The successive functional steps in the application of this  
241 European Standard are shown in Figure 1 and are as follows:
- 242 a) define the Software Requirements Specification and in parallel consider the software architecture.  
243 The software architecture is where the safety strategy is developed for the software and the software  
244 safety integrity level (7.2 and 7.3);
  - 245 b) design, develop and test the software according to the Software Quality Assurance Plan, software safety  
246 integrity level and the software lifecycle (7.4 and 7.5);
  - 247 c) integrate the software on the target hardware and verify functionality (7.6);
  - 248 d) accept and deploy the software (7.7 and 9.1);
  - 249 e) if software maintenance is required during operational life then re-activate this European Standard as  
250 appropriate (9.2).
- 251 A number of activities run across the software development. These include testing (6.1), verification (6.2),  
252 validation (6.3), assessment (6.4), quality assurance (6.5) and modification and change control (6.6).
- 253 Requirements are given for support tools (6.7) and for systems which are configured by application data or  
254 algorithms (8.1).
- 255 Requirements are also given for the independence of roles and the competence of staff involved in software  
256 development (5.1, Annex B and 5.2).
- 257 This European Standard does not mandate the use of a particular software development lifecycle. However,  
258 illustrative lifecycle and documentation sets are given (5.3, Figures 2 and 3 and 7.1).
- 259 Tables have been formulated ranking various techniques/measures against the software safety integrity  
260 levels 1-4. The tables are in Annex A. Cross-referenced to the tables is a bibliography giving a brief  
261 description of each technique/measure with references to further sources of information. The bibliography of  
262 techniques is in Annex D.
- 263 Finally, Annex C provides an example of how the flow between document generation and checking could take  
264 place.



265  
266 **Figure 1 – Illustrative Software Route Map**

## 267 1 Scope

268 1.1 This European Standard specifies the process and technical requirements for the development of  
269 software for programmable electronic systems for use in railway control and protection applications. It is  
270 aimed at use in any area where there are safety implications. These systems can be implemented using  
271 dedicated microprocessors, programmable logic controllers, multiprocessor distributed systems, larger scale  
272 central processor systems or other architectures.

273 1.2 This European Standard is applicable exclusively to software and the interaction between software  
274 and the system of which it is part.

275 1.3 This European Standard is not relevant for software that has been identified as having no impact on  
276 safety, i.e. software of which failures cannot affect the identified safety functions.

277 1.4 This European Standard applies to all safety related software used in railway control and protection  
278 systems, including

- 279 – application programming,
- 280 – operating systems,
- 281 – support tools,
- 282 – firmware.

283 Application programming comprises high level programming, low level programming and special purpose  
284 programming (for example: Programmable logic controller ladder logic).

285 1.5 This European Standard also addresses the use of pre-existing software and tools. Such software  
286 may be used, if the specific requirements in 7.3.4.7 and 6.5.4.13 on pre-existing software and for tools in 6.7  
287 are fulfilled.

288 1.6 Software developed according to any version of this European Standard will be considered as  
289 compliant and not subject to the requirements on pre-existing software.

290 1.7 This European Standard considers that modern application design often makes use of generic  
291 software that is suitable as a basis for various applications. Such generic software is then configured by data,  
292 algorithms, or both, for producing the executable software for the application. The general (sub)clauses 1 to  
293 6.7, 9.1 and 9.2 of this European Standard apply to generic software as well as for application data or  
294 algorithms. The specific subclauses 7.1 to 7.7 apply only for generic software while 8.1 provides the specific  
295 requirements for application data or algorithms.

296 1.8 This European Standard is not intended to address commercial issues. These should be addressed  
297 as an essential part of any contractual agreement. All the clauses of this European Standard will need careful  
298 consideration in any commercial situation.

299 1.9 This European Standard is not intended to be retrospective. It therefore applies primarily to new  
300 developments and only applies in its entirety to existing systems if these are subjected to major modifications.  
301 For minor changes, only 9.2 applies. The determination of the nature and scope of change will be at the  
302 discretion of the assessor. However, application of this European Standard during upgrades and  
303 maintenance of existing software is highly recommended.

## 304 2 Normative references

305 The following referenced documents are indispensable for the application of this document. For dated  
306 references, only the edition cited applies. For undated references, the latest edition of the referenced  
307 document (including any amendments) applies. In case of clauses in the latest edition of the publication  
308 referred that are in contradiction to clauses in this European Standard, this European Standard will overrule  
309 the referenced clauses.

310 EN 50126-1:1999 Railway applications – The specification and demonstration of Reliability, Availability,  
311 Maintainability and Safety (RAMS) – Part 1: Basic requirements and generic process

312 EN 50129:2003 Railway applications – Communication, signalling and processing systems –  
313 Safety related electronic systems for signalling

314 EN ISO 9000 Quality management systems – Fundamentals and vocabulary (ISO 9000)

315 EN ISO 9001 Quality management systems – Requirements (ISO 9001)

316 ISO/IEC 90003:2004 Software engineering – Guidelines for the application of ISO 9001:2000 to computer  
317 software

318 ISO/IEC 9126 series Software engineering – Product quality

## 319 3 Terms, definitions and abbreviations

### 320 3.1 Terms and definitions

321 For the purposes of this document, the following terms and definitions apply.

#### 322 3.1.1 323 assessment

324 process of analysis to determine whether software, which may include process, documentation, system,  
325 subsystem hardware and/or software components, meets the specified requirements and to form a  
326 judgement as to whether the software is fit for its intended purpose. Safety assessment is assessment  
327 focused on but not limited to the safety properties of a system

#### 328 3.1.2 329 assessor

330 entity that carries out an assessment

#### 331 3.1.3 332 commercial off-the-shelf (COTS) software

333 software defined by market-driven need, commercially available and whose fitness for purpose has been  
334 demonstrated by a broad spectrum of commercial users

#### 335 3.1.4 336 component

337 component is a constituent part of software which has well-defined interfaces and behaviour with respect to  
338 the software architecture and design and fulfils the following criteria:

- 339 – it is designed according to “Components” (see Table A.19);
- 340 – it covers a specific subset of software requirements;
- 341 – it is clearly identified and has an independent version inside the configuration management system or is  
342 a part of a collection of components (e. g. subsystems) which have an independent version

#### 343 3.1.5 344 customer

345 entity which purchases a railway control and protection system including the software

- 346 **3.1.6**  
347 **designer**  
348 entity that analyses and transforms specified requirements into acceptable design solutions which have the  
349 required safety integrity
- 350 **3.1.7**  
351 **entity**  
352 person, group or organisation who fulfils a role as defined in this European Standard
- 353 **3.1.8**  
354 **error, fault**  
355 defect, mistake or inaccuracy which could result in failure or in a deviation from the intended performance or  
356 behaviour
- 357 **3.1.9**  
358 **failure**  
359 unacceptable difference between required and observed performance
- 360 **3.1.10**  
361 **fault tolerance**  
362 built-in capability of a system to provide continued correct provision of service as specified, in the presence of  
363 a limited number of hardware or software faults
- 364 **3.1.11**  
365 **firmware**  
366 ordered set of instructions and associated data stored in a way that is functionally independent of main  
367 storage
- 368 **3.1.12**  
369 **generic software**  
370 software which can be used for a variety of installations purely by the provision of application-specific data  
371 and/or algorithms
- 372 **3.1.13**  
373 **implementer**  
374 entity that transforms specified designs into their physical realisation
- 375 **3.1.14**  
376 **integration**  
377 process of assembling software and/or hardware items, according to the architectural and design  
378 specification, and testing the integrated unit
- 379 **3.1.15**  
380 **integrator**  
381 entity that carries out software integration
- 382 **3.1.16**  
383 **pre-existing software**  
384 all software developed prior to the application currently in question is classed as pre-existing software  
385 including
- 386 – COTS (commercial off-the-shelf) and open-source software,
  - 387 – software previously developed but not in accordance with this European Standard
- 388 **3.1.17**  
389 **programmable logic controller**  
390 solid-state control system which has a user programmable memory for storage of instructions to implement  
391 specific functions