UDC 004.4

**M. FUSANI**

***ISTI-CNR, National Research Council, Pisa, Italy***

# EXAMINING SOFTWARE ENGINEERING REQUIREMENTS IN SAFETY-RELATED STANDARDS

*The paper analyses and compares a set of international standards for safety-critical systems whose behavior depends on software. Various types of standards are selected, related to system functional safety in the domains of nuclear plants and transportation (avionics, railway, automotive), as well as software engineering-related standards with the role of reference, focused on process and product quality and on various software technologies. Examination criteria are introduced and discussed, and the standards are compared against such criteria. Some relevant issues are presented, deriving from the comparison, and potential users of the analysis are suggested.*

***Key words:*** *Standards, functional safety, safety-critical systems, software engineering, software quality.*

## Introduction

Increased business and social benefits are expected from products and processes conformity to public standards, as these overcome national boundaries and cross various application domains.

Most often Software Engineering (SE) is invoked, as a transversal technology, by standards for designing and operating different kinds of complex systems. And as a self-standing discipline, SE itself is continuously challenged by the demand for system augmented performance and flexibility coming from industry, boosted by unceasing competition.

This fact has a positive side, in that technology transfer is possible, in principle, from one domain to the other. However, as we note in the following, different standards referring to SE practices just appear to chase each other over time, but in practice they pick from the technology in an unpredictable way.

The purpose of this paper is to examine how the increasingly evolving solutions offered by SE are addressed (or ignored) in Functional Safety-related standards produced and updated at different times and whether or not any advantages could be derived by mutually comparing such standards.

As human life, environment and other valuable resources can be threatened by the (good or bad) behaviour of complex systems, these standards are expected to be the most demanding in terms of balancing intrinsic project and system flexibility due to software with necessary quality, dependability and safety.

We are interested in showing and discussing more an examination approach than the completeness of results. Thus, only some application domains are considered here: aerospace, nuclear power plants, railways and automotive. The benefits of such an analysis would be mainly perceived by: i) standards makers (interested in criteria for practices and techniques selection); ii) standards users (developing organizations, struggling for multiple standards compliance); iii) regulatory organisms, certification bodies and assessors.

In Section 1 some characteristics of the standards are shown and the examining approach is sketched.

In Section 2 a non-exhaustive list of criteria for standards examination and comparison is presented, with concise criteria rationales.

In Sections 3 and 4 the results of the comparison against the criteria are presented and discussed. In Section 5 some conclusions are drawn, that include the improvements of the approach for continuing this research line.

## 1. Overview of Standards and their evolution

As we are dealing with SE in Functional Safety-related Standards (FS_Std), we cannot miss to refer also to the standards concerned with SE only (SE_Std). The relationships between the two categories are one of the objectives of the paper.

### 1.1. FS_Std's

As mentioned in Introduction, we only consider some examples of FS_Std category, that we believe as representative enough of our purposes. Besides the standards related to specific sectors [1]. [2], [3], [4], we add two more general ones, still related to Functional Safety-related systems [5], [6].

We do not consider here for space reasons many other interesting standard-regulated fields such as medical, defence, vessels, chemical plants and others, convinced that further comparison against the criteria applied here can provide even more interesting results, which we leave to the prosecution of this research work.

### 1.2. SE_Std's

We notice that also SE_Std's offer a discontinuous evolution (marked by their publication history) and, even if they seem somewhat more mature in various aspects, they are not in some other. We also realise that it is quite easy to argue that state-of-art in SE is no SE standard: it is constituted by all the research, proposals, case studies and projects reported in scientific papers and books. This general situation is quite a mobile target for our comparison work, so we include some general references, that can be considered as "software technology compendia", which in turn refer to the vast SE literature.

For space reasons again and for sake of generality, we restrict all reference to a limited but comprehensive enough set of items: software process standards [7], [8], [9], product standards [10] and technology compendia [11], [12], [13].

### 1.3. Standard examination approach

Standards will be examined on the basis of comparison criteria. Defining them is partly an arbitrary task, yet we give here some information about the underlying rationale for their selection (Section 3). There is potentially quite a long list of criteria. Here those are shown that mostly struck our attention in a first pass of standard examination.

It should be said that "standards cannot just be read" to be understood, but tried for use. Or made. The author's team used some of them for years at ISTI-CNR in Pisa, for gap-analyses and certification work, and participated in the design of some other. This is no novel work at ISTI-CNR: years ago a similar comparison work was done, with other standards (but including the apparently never-obsolescing DO-178B), with different, more "technical" criteria [14].

### 1.4. How to deal with abstraction levels

One of the most common problems in presenting a technical discipline, especially in computer science, is to uniformly manage abstraction levels when describing a homogeneous list of items or features. One common mistake is mixing levels up: for instance, including implementation strategies and details in product requirements description.

Standards usually do not make evident misuse of levels of abstraction, however sometimes the inter-level boundaries are not clear enough. This is going to be a comparison criterion (Section 2.2).

It should be noticed that a good abstraction level separation helps in a typical problem with standards, that is, the need of keeping a stable enough reference without preventing innovation.

## 2. Criteria for standards examination

### 2.1. Rationale

How can we define such criteria? The purposes are not to judge if a standard (of the SF_Stds's list) is better of fitter to use than another (they belong to different domains, after all), but to check if any of them might gain something useful from another.

Of course, a software-related clause or requirement useful for a defined application domain may be no use or non-applicable for another, so the opportunity of exporting/importing such clause does not depend only on its "modernity". Yet, we see that sometimes less demanding standards recommend more evolved SE practices than other high-criticality norms.

It should be noticed that, if one believes the software as the most critical factor in functional safety-related systems [15], its contribution to system reliability and safety is not quite quantifiable, and still the unsatisfying paradigm "put as much high-tech as you can" is claimed at the higher criticality levels.

Besides, we want to extend the examination beyond the SE aspects, to include some quality issues such as standard usability and ability to assess conformance.

Finally, even if this would be an interesting examination criterion, we are not concerned with political economics motivation for standards.

### 2.2 Criteria and their rationale

Criteria can be derived from the following considerations, or meta-criteria:

1 – common and diverse features (both SE-related and not) of the standards, in terms of contents;

2 - structure, completeness, readability, adequacy to intended use;

3 – contents, in terms of product success assurance (quality may favour success, but not guarantee it);

4 – how peculiarity of the application domain (type of technology, expected impact on safety) is related to the SE (and non-SE) practices prescribed as standard requirements or clauses.

In the following, we list the chosen criteria, together with their rationale. They also are divided into three categories, depending on their nature and on their relationships with SE, and are also ranked according to relevance given by author's experience (this is a quite subjective aspect of the examination, difficult to get rid of). Criteria denoted as CS are more focused in system aspects and qualities of the standard. Criteria denoted as CG are about general contents on software. Criteria denoted as CT are about technical software issues. Meta-criteria relationships are shown for the CS's (within brackets).

CS 1: Arbitrariness of standard interpretation for conformity assessment.

Rationale: The issue depends on how much clear and precise the clauses are in terms of requirements. Not to be confused with degree of freedom in standard compliance by the implementers of requirements (standard users). It puts some problems with certification (meta: 2).

CS 2: Relevance given to product requirements analysis.

Rationale: Requirement phase is where the destiny of all the qualities of the product (typically in operation) is decided (meta: 1, 3).

CS 3: Process conceived as an asset of organized and reusable practices (it survives across projects).

Rationale: Process is not just a collection of practices, but has its own autonomy, human and infrastructure resources, defined inputs and outputs and interacts with other processes. It typically is an organisation's asset (meta: 3).

CS 4: Relevance given to management practices.

Rationale: Management is as essential as techniques for the success of the project (achieving its defined goals) (meta: 3).

CS 5: Means to keep pace with evolving technology.

Rationale: This is not easy in that it involves the policy of the organizations that create and maintain standards, but impacts on: i) separation between standard requirements (what) and implementation (how); ii) introduction of para-standard structures such as standard maintenance records, guidelines, blogs (meta: 3).

CS 6: Relevance given to system theory and system engineering culture.

Rationale: In spite of recent progresses, system theory and system engineering have not much helped each other. Benefits are expected in terms of addressing the systemic aspects of integrity through the understanding of the multiple interactions among system elements, mainly software elements (meta: 3).

CS 7: Relevance given to safety culture.

Rationale: Safety aspects (accidents, hazards and risks) and impact on safety springing from any function, component, level and phase of the project must be understood and sought for by any project worker and stakeholder (meta: 3).

CS 8: Relevance given to human factors.

Rationale: Mutual impact of technology and human aspects (training, psychology, attitude, responsibility) is expected to be balanced at its best (meta: 3).

CS 9: Separating abstraction levels in standard.

Rationale: See Section 1.4 (meta: 2).

CS 10: Introduction of Integrity (SIL) or criticality levels.

Rationale: Although the SIL approach is not new, it is often misunderstood. Good explanation or easy-to-find reference (even cross-standard!) should be provided (meta: 1,3,4).

CS 11: Definition of purpose of the norm and stakeholders.

Rationale: Usually this important information, that is always there, is skipped or overlooked by the reader/user. How to draw attention to it is a presentation feature (meta: 2).

CS 12: Properties of components and systems vs. properties of functions (safety, SIL, reliability).

Rationale: This is a non easy aspect to catch, and needs clear explanation. It is also related with independency between requirements and implementation (meta: 3).

CS 13: Support given to independent assessment or certification.

Rationale: Related to Criterion CS 1. To norm as much as possible of the assessment process helps in giving confidence that the results are repeatable and reproducible, and then the products comparable (meta: 1,2).

CS 14: Relevance given to system operational phase, including human system-related processes.

Rationale: Safety is played in operation, but ways of operating may have requirements, as well the operating environments. This is expected to be included in the standard as well (meta: 1,3).

CG 1: Relevance given to use/reuse of Commercial Off The Shelves (COTS) items and Previously Developed Software (PDS).

Rationale: In-house PDS is quite often an issue. Opportune SE practices about re-use and product lines can get benefits from PDS. Both COTS and PDS must be thoroughly evaluated and classified, and their interfaces totally defined. Risks in changing COTS and PDS must be identified and managed.

CG 2: Relevance given to tool selection and certification.

Rationale: This is a common important feature and its importance is never stressed enough. Less addressed but not less important aspects are relationships between culture and tools, and tools integration problems. Certification should be obtained by an accredited Certification Body. Sometimes, supplier's certification cannot be replaced, but accreditation information should be produced. User's certification cannot give the same confidence and should pair with proven-in-use information.

CG 3: Relevance given to software isolation and over-riding by the hardware.

Rationale: Prevention is quite recommendable. About detection, care must be taken as of over-riding conditions may require equal or higher integrity requirements for the detecting functions.

CG 4: Addressing analysis of software impact into the system and functions traceability.

Rationale: This is a basic problem with software in safety. Whilst isolation (criterion CG 3) includes prevention, impact analysis involves practices and techniques (such as components interaction and interface analysis) to apply any time in the lifecycle.

CG 5: Introducing criteria for selecting software techniques and measures.

Rationale: Techniques should not be just a mention-based list, and not only commented (this would be better but is still rare). Providing elements to judge how and when to use them, and requesting explicit motivation for it is a success factor.

CG 6: Relevance given to configuration data.

Rationale: As important as software, they are perhaps easier to deal with, but should have their own documented lifecycle.

CG 7: Addressing connections between Software and System.

Rationale: This is a borderline where first of all competencies of system engineers and software engineers should overlap and mutually corroborate.

CT 1: Relevance given to specific Formal Methods (FM).

Rationale: Just requesting "formal methods" does not give the user much help. Criteria for method selection should be provided. FM is now an affordable technology, also from the cultural point of view, and is supported by automation.

CT 2: Introducing Model Checking**.**

Rationale: This FM category that allows to prove properties on a modelled software system (such as reachability of defined states and critical races) is much more powerful that most static analyses.

CT 3: Introducing model based development.

Rationale: Mostly used in practice, hardly appears in standards. Not a real FM, can be connected with FM-related tools.

CT 4: Relevance given to software requirements analysis.

Rationale: A must (criterion CS 2). Can and should be supported by tools. Implicit or explicit quality model for requirements is expected.

CT 5: Relevance given to verification and Testing.

Rationale: This is not only common issue but also quite a broad area and it is expected to be covered in terms of various categories of verification including test procedures and tools, test levels, data definition, types of tests, type of coverage, testing techniques, testing process.

CT 6: Relevance given to diversity.

Rationale: Diversity needs careful planning, execution and verification to avoid that it increases complexity. Almost unavoidable when there is hardware redundancy. It is costly.

CT 7: Introducing reliability testing.

Rationale: Estimating error rates on the basis of statistics over test results would be much useful. There has been a lot of effort over many years, and some results are promising. It requires culture and maturity in the organisation.

## 3. Examining and comparing standards in the light of defined criteria

In this ongoing research, we cross the selected criteria with all the selected standards (both FS_Std's and SE_Std's). For each criterion, we examine how it is met by each standard. We want to make it clear that this is no merit judgment, but just detection: in fact, there can be good reasons why some expectations are not achieved completely.

In Table.1 the results of a first-step examination are shown, where each of the criteria (whose definition was re-worded for space reasons) can be seen as a factor or a variable against which contents and properties of the standards can be compared. Ordinal scoring values (ranging 0 to 4) show how the standard is sensible to the variables representing the criteria (4 is the highest sensibility).

The values have been assigned on the basis of the quantity of information produced per criterion. Contents are, of course, more important that quantity and have been also taken into account. Hyphens (" – ") mean either that the criterion is not applicable (such as for IEC 60880, that addresses, by purpose, only some SE practices), or that has not been applied yet (activity is progressing: only a subset of the more than 300 statements that would result from the complete crossing have been resolved).

What is important, analysing the table by rows, is to enlighten those criteria that have higher rank in the list and *greater variability* across the standards. This would mean that important features could be migrated from one standard to another, or that some standard was developed at a different maturity stage of the particular practice or technology referred by the criterion.

It is more interesting, or more useful, to analyse (still by rows) the relationships between FS_Std's and SE_Std's. The overlap we find here has a different meaning than that found within the FS_Std's only.

## 4. Discussion

We may expect that the most recently updated standards adopt the most innovative techniques. This is only partially true.

The findings seem to show less sensitivity to standard updates. Application domains are likely responsible of the bigger difference.

It can be seen from the overall picture that some aspects are addressed with similar emphasis (and amount of information), such as testing. In our analysis, this is not very relevant, as no standard has to give much to others.

There are of course differences (for instance, some standards, in their techniques and measures parts, do not recognize the value of input/output test data), but, in this first phase of the research, we keep our value scale at a moderate level of detail (a sort of filtering, to make evident only the most interesting variations).

Table 1

Results of the examination

| | CRITERIA | | | IEC 60880-2 | DO-178B | EN 50128/9 | ISO 26262-6/8 | IEC 61508-2/3 | CMMI +Safe | ISO/IEC 12207 | CMMI | ISO/IEC 15504 | ISO/IEC 9126 / 14102 | SWEBOK | ISTQB Syllabus | SEI Process Framework |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | keywords | | rank | | | | | | | | | | | | | |
| System/Software/Std Quality | CS 1 | interpretation for conformity | 4 | 2 | 3 | 3 | 3 | 3 | 1 | 2 | 3 | 3 | 2 | - | - | - |
| | CS 2 | requirements analysis | 4 | - | 3 | 2 | 4 | 3 | 3 | 3 | 4 | 4 | - | 2 | - | 3 |
| | CS 3 | process concept | 4 | - | 4 | 1 | 3 | 1 | 4 | 4 | 4 | 4 | - | 4 | 2 | 4 |
| | CS 4 | management | 3 | 2 | 3 | 3 | 4 | 3 | 3 | 2 | 4 | 4 | - | 2 | 2 | 2 |
| | CS 5 | evolving technology | 3 | 2 | 2 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 2 | 4 | 3 | 4 |
| | CS 6 | system engineering | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | - | 1 | 2 |
| | CS 7 | safety culture | 3 | 1 | 3 | 3 | 3 | 3 | 4 | - | - | - | - | - | - | - |
| | CS 8 | human factors | 3 | - | 1 | 1 | 1 | 2 | 3 | 1 | 2 | 2 | - | 2 | - | 3 |
| | CS 9 | abstraction levels | 3 | 3 | 4 | 2 | 3 | 2 | 2 | 3 | 2 | 4 | - | 3 | 2 | 4 |
| | CS 10 | Integrity levels | 2 | - | 4 | 4 | 4 | 4 | 1 | - | - | - | - | - | - | - |
| | CS 11 | purpose and stakeholders | 2 | 2 | 3 | 3 | 3 | 3 | 2 | - | - | - | - | - | - | - |
| | CS 12 | safe components vs. functions | 2 | 2 | 2 | 3 | 3 | 3 | - | - | - | - | - | - | - | - |
| | CS 13 | independent certification | 1 | 2 | 4 | 2 | 3 | 3 | 1 | 2 | 3 | 3 | 2 | - | - | - |
| | CS 14 | operational phase | 1 | 3 | 3 | 2 | 4 | 2 | 1 | 2 | 2 | 2 | 2 | - | - | - |
| Software General | CG 1 | COTS / PDS | 4 | 4 | 3 | 2 | 3 | 2 | - | - | - | - | 3 | - | 3 | 1 |
| | CG 2 | tools selection / certification | 4 | 4 | 2 | 3 | 4 | 3 | - | - | - | - | 4 | - | - | - |
| | CG 3 | SW isolation | 3 | 3 | 4 | 2 | 2 | 3 | - | - | - | - | - | - | - | - |
| | CG 4 | impact analysis / traceability | 3 | 4 | 3 | 3 | 3 | 3 | - | - | 2 | 2 | - | - | - | - |
| | CG 5 | technique selection criteria | 2 | - | 2 | 3 | 1 | 3 | 2 | - | 1 | 1 | - | 3 | - | 1 |
| | CG 6 | configuration data | 1 | - | 2 | 4 | - | - | - | - | - | - | - | - | - | - |
| | CG 7 | SW - System borderline | 1 | - | 4 | 2 | 3 | 3 | - | - | 2 | 1 | - | - | - | - |
| Software techniques | CT 1 | Formal Methos | 4 | 1 | 1 | 1 | 1 | 1 | - | - | - | - | - | 2 | - | - |
| | CT 2 | Model checking | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| | CT 3 | model based developent | 3 | - | - | - | 3 | - | - | - | - | - | - | - | - | - |
| | CT 4 | requirements analysis | 3 | - | 3 | 2 | 3 | 2 | 3 | 3 | 4 | 4 | - | 4 | - | - |
| | CT 5 | verification and testing | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 4 | 4 | - |
| | CT 6 | diversity | 2 | 4 | 3 | 2 | 2 | 2 | - | - | - | - | - | - | - | - |
| | CT 7 | reliability testing | 1 | - | 1 | - | - | - | 1 | - | - | - | - | 4 | 2 | - |
| | | total score | | 44 | 73 | 61 | 70 | 63 | 40 | 31 | 42 | 43 | 19 | 34 | 19 | 24 |

Reading by columns, what strikes the attention is that FS_Std DO-178B, yet 18-year old, still gets a relevant score.

This standard was thought of with certification (basically, aircraft certification) in mind, and the overall system (the product) is always in sight throughout its clauses, even when software components are described.

The process concept (CS 3), independently developed by DO-178B and other SE standards (such as ISO/IEC 12207), should be better adopted by sector standards, and in any case it should be harmonised with the SE_Std's.

This is really important because many companies, to be conformant, feel that they must maintain, with difficulty, different kinds of documents and skills for the same task.

Cross-fertilization among standards of different application domains would not necessarily be inhibited by cost-barriers built by more critical sectors: automation is going to enter, affordably and effectively in the software process and, as pointed out in Introduction, there is just one Software Engineering supporting all kinds of systems, including Functional Safety-Related Systems.

Techniques such as Formal Methods (FM) and Reliability Testing are not much addressed, but there is evidence that the former will gain much and explicit techniques such as Model Checking will appear in the next editions, again due to more feasible and affordable automated practices. A summary of current initiatives about FM can be found in [16]. The latter technique, proposed in literature since long [17], would be highly valuable, but practitioners do not have still much confidence in it: also proving its efficacy would be costly.

In general, we see that not necessarily the most critical and sensitive fields get the most innovative stuff. Standard makers working groups are heterogeneous: as people from academia get in there to innovate, people from industry, with good reasons, tend to be conservative, excepting when it is matter of already acquired technology.

## Conclusions and Improvements

A set of international standards for safety-critical systems have been compared in the light of defined criteria, mostly oriented to analyse how the standards make use of Software Engineering (SE) techniques. To do so, some standards in the SE field have been also selected and analysed according to the same criteria.

Comparison results have been discussed, showing that some safety critical-related standards would benefit by importing practices and techniques from other similar standards and from SE standards with no higher cost, and that there are cases where SE standards themselves could be improved. What has been presented is basically a method, and this method is going to be improved and used in an ongoing research work. In fact, a sort of finer evaluation (still in ordinal scales) will be adopted and some light factor analysis will be used, in order to measure the variability that has been qualitatively shown, to look for factors aggregates and to indicate areas in which practice migration could be made more effective.

## References

1. *IEC 60880-2, Software for Computers in the Safety Systems of Nuclear Power Stations – Software aspects of defence against common cause failures, use of software tools and of pre-developed software / IEC, 2000.*

2. *RTCA DO-178B, Software considerations in airborne systems and equipment certification / RTCA, 1992.*

3. *CENELEC - EN 50128 – Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems / CENELEC, 2002.*

4. *Bellotti M. Seminar on ISO/CD 26262 Road vehicles – Functional safety - Part 6: Product development: software level – Part 8: Supporting processes [Электронный ресурс] / M. Bellotti, 2008. – Режим доступа к ресурсу: http://www.automotive-spin.it/download.php.*

5. *IEC 61508, Functional safety of electrical/electronic/programmable electronic safety-related systems, Part 3, Software requirements / IEC, 1997.*

6. *SEI A Safety Extension to CMMI-DEV, V1.2 TECHNICAL NOTE CMU/SEI-2007-TN-006 / SEI Carnegie-Mellon, 2007.*

7. *ISO/IEC 12207 FDAM Information technology – Software life cycle processes / ISO/IEC, 1995.*

8. *Chrissis M. CMMI: Guidelines for Process Integration and Product Improvement / M. Chrissis, M. Konrad, S. Shrum. - Addison-Wesley Professional, 2006.*

9. *ISO/IEC 15504-2 Information technology -- Process assessment -- Part 2: Performing an assessment / ISO/IEC, 2003.*

10. *ISO/IEC 9126-1 Software engineering- Product quality- Part 1: Quality model / ISO/IEC, 2001.*

11. *SWEBOK Guide to the Software Engineering Body of Knowledge / A project of the IEEE Computer Society Professional Practices Committee / IEEE, 2004.*

12. *International Software Testing Qualification Board - ISTQB Syllabus / ISTQB, 2007 (http://www.istqb.org/download.htm).*

13. *SEI A Process Research Framework / The International Process Research Consortium / SEI Carnegie-Mellon, 2006.*

14. *Mazzanti F. Assessment of the Safety of Hazardous Industrial Processes in the Presence of Design Faults / F. Mazzanti, L. Strigini. - SHIP Project, 1993.*

15. *Leveson N. System Safety Engineering: Back To The Future / N. Leveson, 2002 (http://sunnyday.mit.edu/).*

16. *Ponsard C. From Rigorous Requirements Engineering to Formal System Design of Safety-Critical*

*Systems [Электронный ресурс] / C. Ponsard, P. Massonet, G. Dallons. - ERCIM NEWS, 2008. Vol. 75. - P. 22-23. – Режим доступа к ресурсу: http://ercim-news.ercim.org/content/blogcategory/0/699/.*

*17. Frankl P. Choosing a testing method to deliver reliability / P. Frankl, D. Hamlet, B. Littlewood, L. Strigini. - International Conference on Software Engineering, 1997. - P. 68-78.*

*Поступила в редакцию 2.02.2009*

**Рецензент:** д-р техн. наук, доцент, проф. кафедры О.В. Поморова, Хмельницкий национальный университет, Хмельницкий, Украина.

## АНАЛИЗ ТРЕБОВАНИЙ К РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В СТАНДАРТАХ ПО БЕЗОПАСНОСТИ

*М. Фузани*

В статье анализируются и сравниваются международные стандарты для систем с особыми требованиями по безопасности, поведение которых зависит от программного обеспечения. Выбраны различные типы стандартов, относящиеся к функциональной безопасности системы в сфере атомных станций и транспортных средств (авиация, железная дорога, автомобилестроение), а также стандарты, относящиеся к разработке программного обеспечения, сосредоточенные на процессе и качестве продукта и на различных технологиях программного обеспечения. Приведен и обоснован критерий анализа и произведено сравнение стандартов согласно этому критерию. Представлены некоторые важные результаты сравнения и предложены потенциальные пользователи этого анализа.

**Ключевые слова:** стандарты, функциональная безопасность, системы с особыми требованиями к безопасности, разработка программного обеспечения, качество программного обеспечения**.**

## АНАЛІЗ ВИМОГ ДО РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ У СТАНДАРТАХ З БЕЗПЕКИ

*М. Фузані*

У статті аналізуються та порівнюються міжнародні стандарти щодо систем з особливими вимогами з безпеки, поведінка яких залежить від програмного забезпечення. Обрані різні типи стандартів, що стосуються функціональної безпеки системи у сфері атомних станцій та транспортних засобів (авіація, залізниця, автомобілебудування), а також стандарти, що відносяться до розробки програмного забезпечення та зосереджені на процесі і якості продукту та різних технологіях розробки програмного забезпечення. Приведено і обґрунтовано критерій аналізу та проведено порівняння стандартів згідно цього критерію. Представлені деякі важливі результати порівняння та запропоновані потенціальні користувачі цього аналізу.

**Ключові слова:** стандарти, функціональна безпека, системи з особливими вимогами до безпеки, розробка програмного забезпечення, якість програмного забезпечення.

**Марио Фузани –** канд. техн. наук, научный сотрудник Центра оценки программного обеспечения и систем, Пиза, Италия, e-mail: mario.fusani@isti.cnr.it