Classification of Diversity for Dependable and Safe Computing

Viacheslav Frolov[1][0000-0002-5860-7193], Oleksandr Frolov[2][0000-0002-0230-0790],
Vyacheslav Kharchenko[3][0000-0001-5352-077X]

[1-3]National Aerospace University named after N. Y. Zhukovsky "KhAI", Kharkiv, Ukraine
v.frolov@csn.khai.edu,o.frolov@csn.khai.edu,
v.kharchenko@csn.khai.edu

Abstract. The article deals with generalization of diversity and version redundancy application to protect security and safety critical systems. The main problem for such systems is the occurrence of common cause failures (CCF). It is shown that as dependable computing is combining of reliable, safe and secure computing a principle of diversity can be applied to minimize risks of CCF and assurance of requirements to all attributes. It is reported about need to respond to new technologies and challenges in this area by improvement exist classifications of diversity. It is given the facet-hierarchical representation of the most representative schemes of diversity. The united classification of types of diversity from different domains is proposed. The classification scheme is based on the NUREG-7007. A possibility to assess and assure security and safety by using the resulting classification is discussed for clouds systems and other technologies.

Keywords: diversity, NUREG, facet-hierarchical classification, united classification, common cause failures, critical systems.

1 Introduction

1.1 Motivation

Development and implementation of dependable software and hardware components becomes more and more important for safety and security critical systems. The main problem for such applications is to reduce the risk of common cause failures (CCF). CCF is coincidental failure of two or more structures, systems or components is caused by any latent deficiency from design or manufacturing, from operation or maintenance errors, and which is triggered by any event induced by natural phenomenon, plant process operation, or action caused by man or by any internal event in the instrumentation and control system [1-3].

One of the most effective solutions to the CCF problem is diversity, which makes each version of the system unique, that means the probability of occurrence of CCF is minimized [2]. Diversity is a principle in instrumentation systems of sensing different parameters, using different technologies, using different logic or algorithms, or using different actuation means to provide several ways of detecting and responding to a significant event [2, 3].

Thus, solving the same problem in different ways, for example using different programming languages or different development teams, you can avoid system failure due to one error. Since in this case the faults will be various in each version, which means that the probability of CCF is significantly reduced.

Given the importance of effective implementation of diversity, there is a need to evaluate and select the most appropriate type of diversity. However, with the development of information technology, such task becomes more and more difficult, as the question arises as to how to use diversity and how to evaluate the security of multi-version systems when using such technologies. In addition, designers are trying to apply new technologies to provide diversity. This leads to a difficult choice of the type of diversity for a particular application.

To make this choice, a clear, ordered structure is needed to narrow the search circle. There are a large number of different classifications that represent various aspects of diversity [2-9]. Despite this, over time emerges a need to update and generalize them with considering of new technologies and challenges.

1.2 State of the art

Currently, there are many different classifications of diversity, each of which deals with a narrow aspect of the security of a particular system. However, such a deep detail prevents to look at the situation as a whole. Consequently, it have to combine the most well-known classifications, which cover a sufficiently large number of types of diversity.

The most representative on this topic are following works. The technical report of the diversity of critical systems in the nuclear power industry NUREG-7007 [2], which in fact got status of standard, considers the protection of both software and hardware components. At the same time, software diversity is limited to N-version programming. The paper “The Multiple Faces of Software Diversity ...” [4] structures the research devoted to software diversity. It is the most complete overview of works on software diversity, although it does not contain its exhaustive classification. In addition, there is the article describing where and when it is possible to apply diversity to protect the software “SoK: Automated Software Diversity” [5]. Besides, authors analyze automated techniques of software diversity beyond the scope of N-version programming and suggest the classification of diversity based on software life cycle. However, it is not enough to describe all types of diversity.

There is a number of works, which clarify some aspects of diversity. The monograph [6] considers diversity to assure software fault tolerance. It is a classic work on this topic, although it is somewhat outdated at the moment. The article [7] offers a model of multi-version (n, m) – version systems that details the structure of N-version programming. An extension of the classification [2] was proposed in [8].

Hence, there are a lot of classifications and applications of diversity in safety and security critical domains during last twenty years. However, there is not general classification of diversity and techniques to join different version redundancy types.

1.3 Goals and structure

The goal of the paper is to develop the unified classification of diversity. For this, we first consider the basic principles of the construction of classifications and describe the general approach to the representation of diversity in the facet-hierarchical structure (Section 2). Next, a plan for combining classifications is disclosed and the process of building a unified classification of diversity is described (Section 3). Results of the united classification development are presented (Section 4). Last section concludes and presents future steps of research.

2 Principles of classification

2.1 Type of classifications for diversity

The main types of classifications are faceted, hierarchical and facet-hierarchical (FIC) [10]. The basis of the facet classifications is the emphasizing of classification features in the subject domain under consideration, which are called facets. All terms from this area attribute to a particular facet.

The hierarchical classification method consists in dividing the whole set of terms from the subject domain into groups, which are further divided into subgroups, and these in turn can be further divided. Thus, subordination is established between the classification groups, which has the form of a hierarchy.

FIC consists of taxons and classification attributes. First, a set of objects form taxons, which are divided into groups according to classification features, which in turn are arranged in a certain hierarchy. The taxons themselves are also divided into subgroups, thus forming a subordination between taxons.

2.2 General approach

Depending on the type of classification, the approach to its construction is somewhat different, but there is a general principle for constructing any classification.

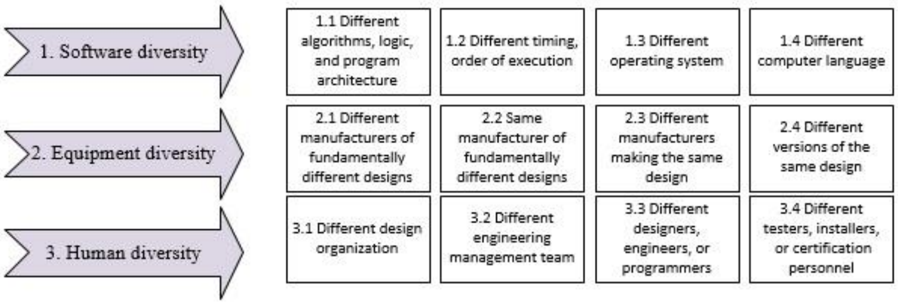
First of all, the formation of a set of studied objects is occurred. Each element of which must be deterministic and should be given a semantic description. After that, the set is divided into subsets according to certain attributes, which are unique for each subset. At the same time, depending on the type of classification, subordination is established between subgroups or this is not required.

Thus, works on diversity that contain classification schemes can be analyzed. These schemes are presented in the form of FIC. After that, they have been combined into a unified classification to cover the overall picture, taking into account modern technologies.

3 Integrating of classification schemes

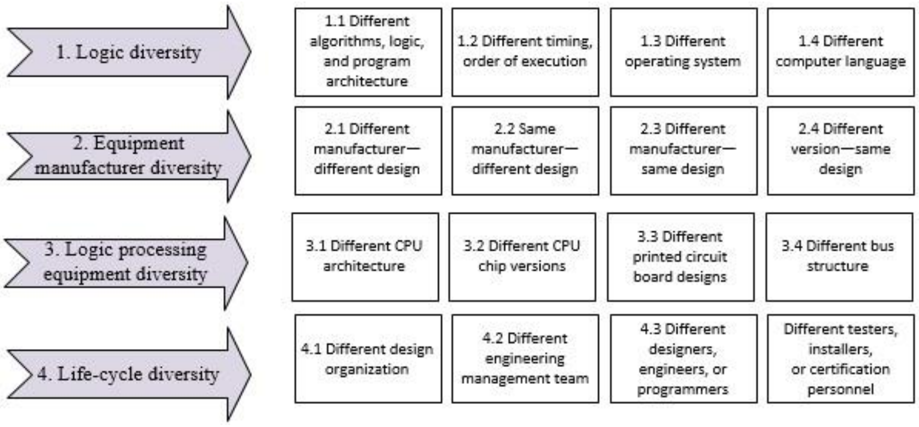
3.1 Facet classifications

The example of extending faceted classifications is obtaining NUREG-7007 [2] from NUREG-6303 [1] by splitting one facet (Equipment diversity) into two (Equipment manufacturer diversity and Logic processing equipment diversity), as well as changing the semantic content of the other two facets (Software diversity and human diversity). Software diversity has come to be called the logic diversity and human diversity, respectively, is called the life-cycle diversity. Thus, the main changes in these classifications are shown on figures 1 and 2 according with technique described in [10].



1. Software diversity	1.1 Different algorithms, logic, and program architecture	1.2 Different timing, order of execution	1.3 Different operating system	1.4 Different computer language
2. Equipment diversity	2.1 Different manufacturers of fundamentally different designs	2.2 Same manufacturer of fundamentally different designs	2.3 Different manufacturers making the same design	2.4 Different versions of the same design
3. Human diversity	3.1 Different design organization	3.2 Different engineering management team	3.3 Different designers, engineers, or programmers	3.4 Different testers, installers, or certification personnel

Fig. 1. NUREG-6303 based diversity classification.



1. Logic diversity	1.1 Different algorithms, logic, and program architecture	1.2 Different timing, order of execution	1.3 Different operating system	1.4 Different computer language
2. Equipment manufacturer diversity	2.1 Different manufacturer—different design	2.2 Same manufacturer—different design	2.3 Different manufacturer—same design	2.4 Different version—same design
3. Logic processing equipment diversity	3.1 Different CPU architecture	3.2 Different CPU chip versions	3.3 Different printed circuit board designs	3.4 Different bus structure
4. Life-cycle diversity	4.1 Different design organization	4.2 Different engineering management team	4.3 Different designers, engineers, or programmers	Different testers, installers, or certification personnel

Fig. 2. NUREG-7007 based diversity classification.

3.2 Hierarchical classifications

The example of extending of a hierarchical classification is the combination of software diversity (SoK: Automated Software Diversity and (n, m) – version systems).

The idea of extending is to divide the Implementation stage into 4 stages (Requirements development, Design, Coding and debugging, Integration).

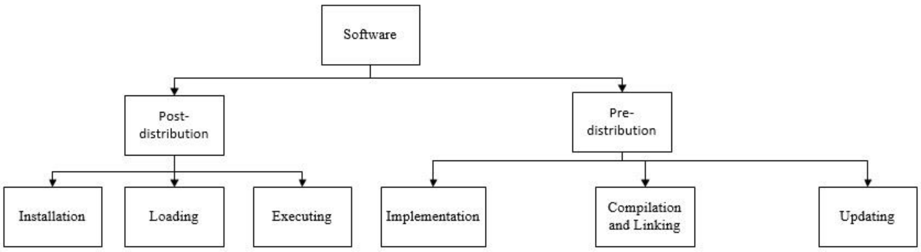


Fig. 3. Automated Software Diversity based software life-cycle.

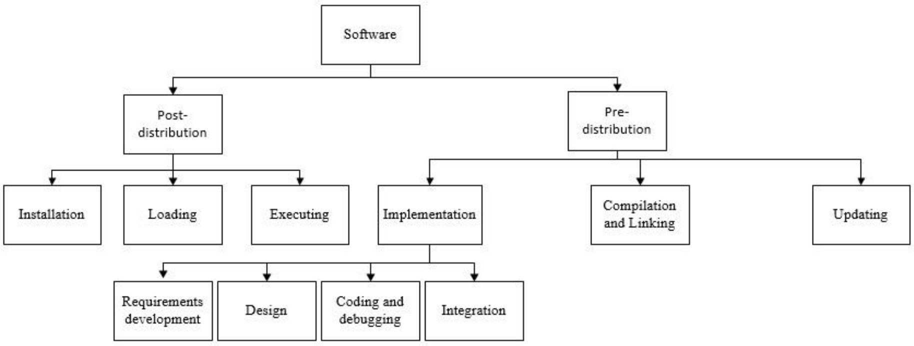


Fig. 4. Automated Software Diversity and (n,m)- version systems based software life-cycle.

3.3 Facet-hierarchical classifications

Taxonomic structure is set, consisting of three elements [10]:

$$S\ \{A, T, \Psi\}, \tag{1}$$

where A — a set of classification attributes $A = \{A_i\}_{i=1}^{n}$, T — a set of taxons $T = \{t_i\}_{i=1}^{n}$, Ψ — relationship between elements $A_i \in A$ and $T_i \in T$: i.e.: A Ψ T.

An example of the taxonomic structure is the facet classification discussed above. The main feature of which is the independent of all facets.

Since the types of diversity are not equivalent, but are subordinate to each other, it is advisable to use faceted hierarchical structure. In which the classification attributes and taxons have a hierarchical form. Thus, a hierarchical component is added to equation (1).

Facet-hierarchical structure is set, consisting of three elements [10]:

$$S_H = \{A_H, T_H, \Psi_H\}, \tag{2}$$

where AH = $\{A_{Hi}\}_{i=1}^{n}$ — a set of classification attributes in hierarchical structure, T_H = {... $\{t_{ij...k}\}$ ...} — a set of taxons in hierarchical structure, ΔT = $\{t_i\}$ — subset of taxons in hierarchical structure, which appropriate to classification attribute, $\Psi: \forall A_{Ii} \leftrightarrow \Delta T_{Ii} \subset T_I$ —relation between taxonomic classification attributes and a variety of taxons in a hierarchical structure.

The example of applying facet-hierarchical classification is construction the main part of software diversity from NUREG-7007 [2], Automated Software Diversity [5] and (n,m)- version systems [7]. The idea is to distribute the techniques of software diversity according to the stage of the life cycle to which they apply. This example can be represented as a matrix showing the relationship between attributes and taxons. The result is presented in Table 1.

Table 1. FIS matrix representation.

Classification attributes	Taxon 1	Taxon 2	Taxon 3	Taxon 4	Taxon 5	Taxon 6
1. Software diversity	Post-distribution	Pre-distribution				
1.1 Software life-cycle	Installation	Loading	Executing	Implementation	Compilation and Linking	Updating
1.1.1 Stages of implementation	Requirements development	Design	Codding and debugging	Integration		
1.2 Implementation and Installation techniques	In-place diversification	N-self checking programming	Recovery Blocks	N-Version programming		
1.2.1 Types of N-Version programming	Different algorithms, logic	Different timing and/or order of execution	Different runtime environment	Different functional representation		

Then a hierarchy is built up between taxons according to their correlation. As a result, a scheme describing some types of software diversity has been obtained (see Figure 5). Further, using the described procedure, the program diversity has been supplemented. In particular, for the stages of Compilation and Linking, Updating, the techniques that are used on them have been described.

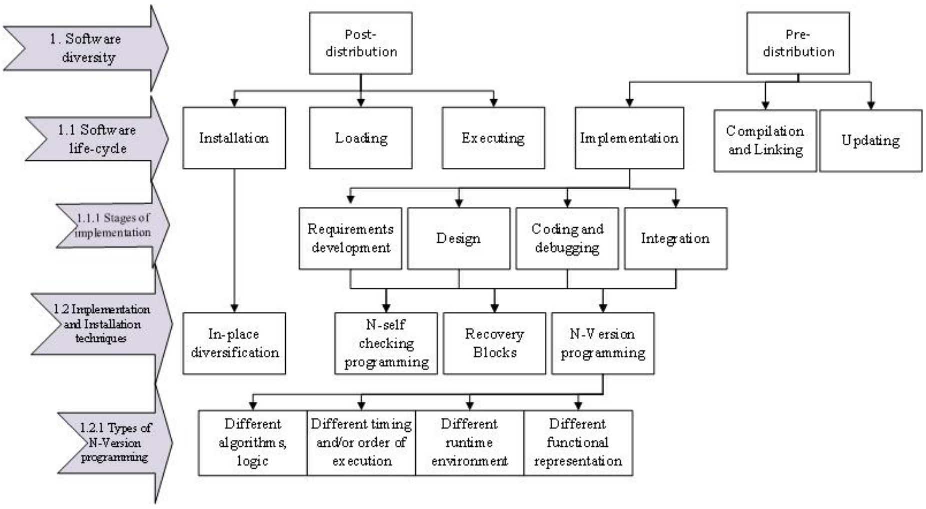

1. Software diversity
1.1 Software life-cycle
1.1.1 Stages of implementation
1.2 Implementation and Installation techniques
1.2.1 Types of N-Version programming
Post-distribution
Pre-distribution
Installation
Loading
Executing
Implementation
Compilation and Linking
Updating
Requirements development
Design
Coding and debugging
Integration
In-place diversification
N-self checking programming
Recovery Blocks
N-Version programming
Different algorithms, logic
Different timing and/or order of execution
Different runtime environment
Different functional representation

Fig. 5. The main part of software diversity.

4 Case study

4.1 Basic classification schemes and features of integration

To develop a generalized classification, the schemes described in [2-9] have been selected. After that, they have been systematized into groups: Software, Hardware, Information management system. Human diversity, which can be used for any system and consists in using different teams of designers, developers and verifiers for each version, is considered separately.

The above classifications have been combined into a unified semantic facet-hierarchical structure (2). Since the process of combining classifications is not fully formalized in terms of the semantic component, expert support was used. To build a unified classification, the following procedure can be used:

- Presenting the selected classifications in the form of FIS;
- Preparing the initial data for the united (classification attributes and taxons);
- Establishing the equivalences between attributes;
- Establishing the taxon equivalences for each attributes;
- Clarifying the hierarchies between attributes and between taxons;
- Developing the final version of the classification.

4.2 Integrated diversity classification scheme

Integrated diversity classification scheme presented in the figure 6. The scheme demonstrates a facet-hierarchical representation of diversity, where the top level contains such systems as software, hardware, information management. It also includes human diversity, which can be used for any type of system.

The general principle of construction of the scheme is the life cycle of diversified systems .This is special clearly shown for software, where the techniques providing diversity have been assigned to the stage of the life cycle to which they correspond. Since the implementation stage is the most sought-after to provide protection, it was described in more detail. Then attention was paid to fundamentally different approaches to the diversity of software, such as automated and managed diversity. The main difference of them is the origin of the diversity. This use of naturally established features of software, or artificial [11] introduction of a diversity.

The diversity of hardware has been structured according to the features of its operation and the most important components such as equipment, design, functioning and signal. In particular, the diversity of equipment is divided into two groups: equipment manufacturer and equipment processing logic. The possibility of using the principle of diversity at the network level has also been described.

4.3 Applying diversity classification scheme in cloud computing

One of the areas in which this classification can be used is the development of applications hosted in the cloud [12]. In this case, the first thing to do is to choose at what stage of the life cycle to apply the principle of diversity. After evaluating the most acceptable stage in terms of costs and security level, it is necessary to get acquainted with the techniques which are used on it and select the appropriate one.

Further, the decision is made on what type of application will be monolithic or consisting of micro-services. In accordance with this, it is necessary to use techniques of either managed diversity or automated one. This decision will also affect whether different teams of developers, testers and others will be used.

Since the cloud provider ensures the hardware, its diversity does not apply. However, it is possible instead diversify the providers themselves or the geographical location of their servers.

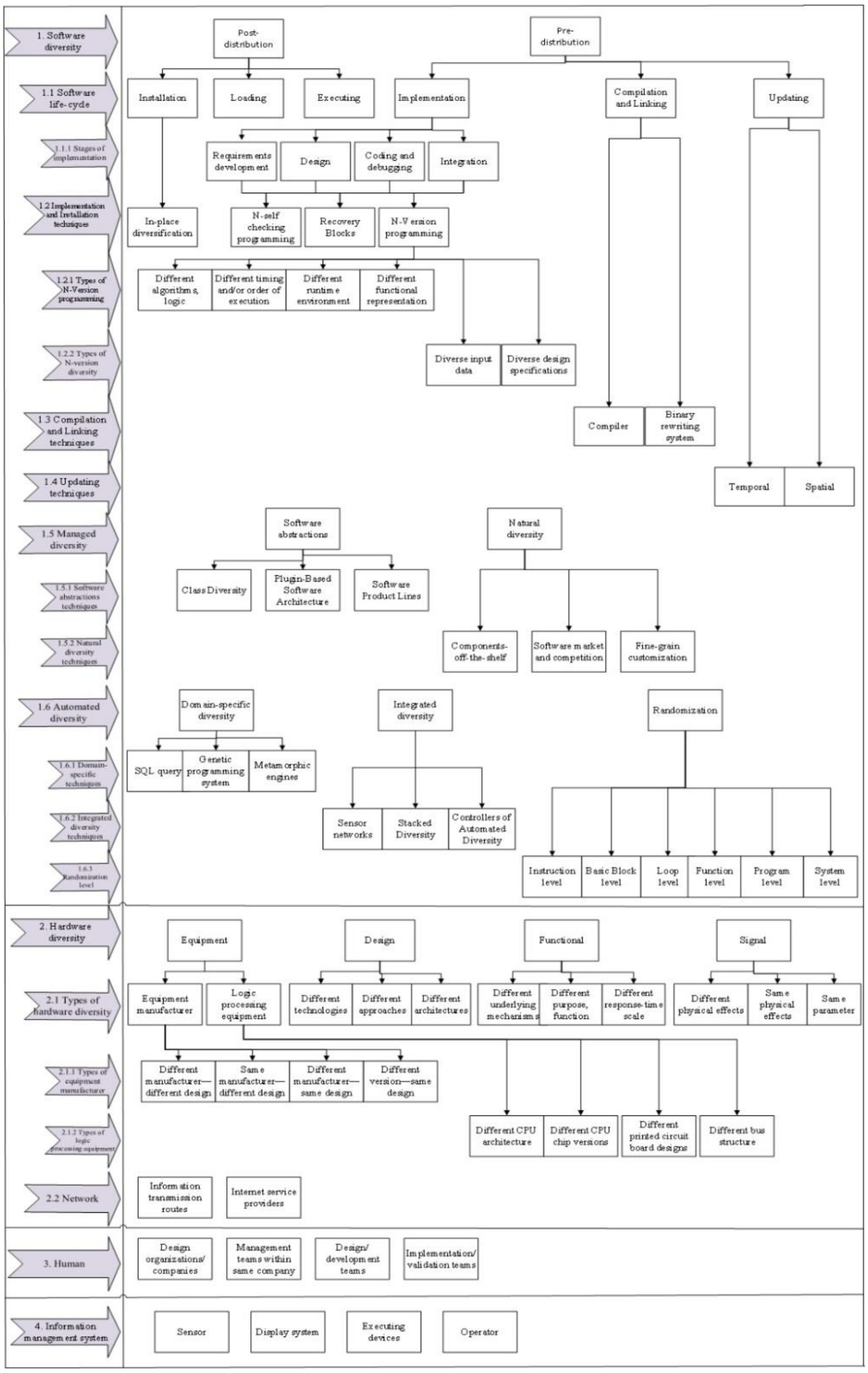

1. Software diversity
1.1 Software life-cycle
1.1.1 Stages of implementation
1.2 Implementation and Installation techniques
1.2.1 Types of N-Version programming
1.2.2 Types of N-version diversity
1.3 Compilation and Linking techniques
1.4 Updating techniques
1.5 Managed diversity
1.5.1 Software abstractions techniques
1.5.2 Natural diversity techniques
1.6 Automated diversity
1.6.1 Domain-specific techniques
1.6.2 Integrated diversity techniques
1.6.3 Randomization level
2. Hardware diversity
2.1 Types of hardware diversity
2.1.1 Types of equipment manufacturer
2.1.2 Types of logic processing equipment
2.2 Network
3. Human
4. Information management system
Post-distribution
Pre-distribution
Installation
Loading
Executing
Implementation
Compilation and Linking
Updating
Requirements development
Design
Coding and debugging
Integration
In-place diversification
N-self checking programming
Recovery Blocks
N-Version programming
Different algorithms, logic
Different timing and/or order of execution
Different runtime environment
Different functional representation
Diverse input data
Diverse design specifications
Compiler
Binary rewriting system
Temporal
Spatial
Software abstractions
Natural diversity
Class Diversity
Plugin-Based Software Architecture
Software Product Lines
Components-off-the-shelf
Software market and competition
Fine-grain customization
Domain-specific diversity
Integrated diversity
Randomization
SQL query
Genetic programming system
Metamorphic engines
Sensor networks
Stacked Diversity
Controllers of Automated Diversity
Instruction level
Basic Block level
Loop level
Function level
Program level
System level
Equipment
Design
Functional
Signal
Equipment manufacturer
Logic processing equipment
Different technologies
Different approaches
Different architectures
Different underlying mechanisms
Different purpose, function
Different response-time scale
Different physical effects
Same physical effects
Same parameter
Different manufacturer—different design
Same manufacturer—different design
Different manufacturer—same design
Different version—same design
Different CPU architecture
Different CPU chip versions
Different printed circuit board designs
Different bus structure
Information transmission routes
Internet service providers
Design organizations/ companies
Management teams within same company
Design/ development teams
Implementation/ validation teams
Sensor
Display system
Executing devices
Operator

Fig. 6. United classification scheme.

Currently, web-services are actively developing. Their main components are application layer, web server, apps server, operating system and database management system. Web-services can be developed using cloud computing. In this case, the application will consist of components-off-the-shelf [13]. Thus, we can use natural diversity. It is also necessary to use the principle of diversity on the network layer. Using the methodology of faceted hierarchical structures and the constructed united classification, you can choose which type of diversity to apply for cloud computing. For such systems, the classification will take the form presented in figure 7.

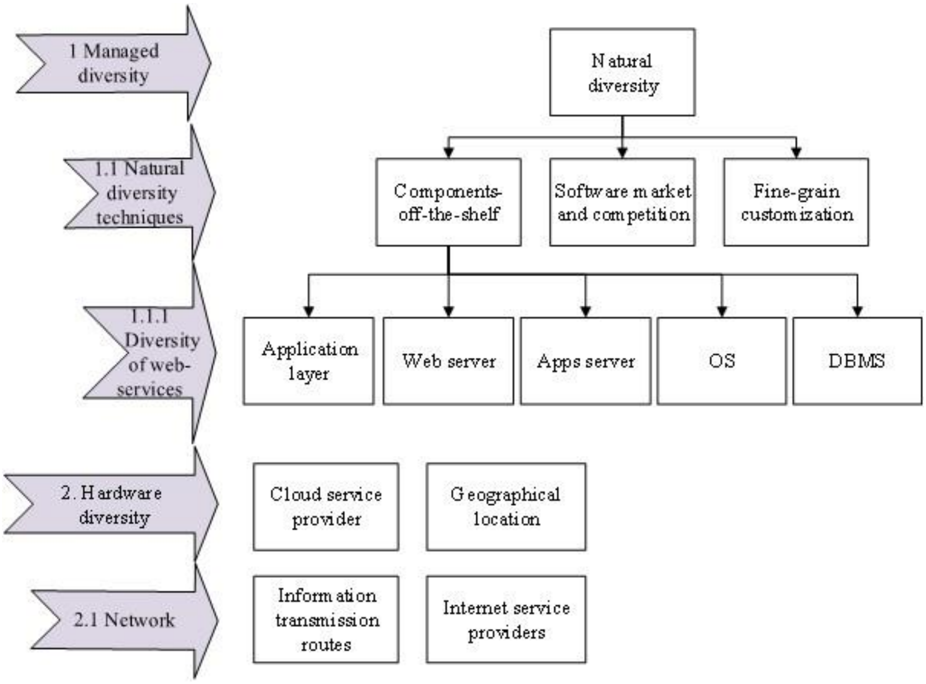


Fig. 7. Application of classification in cloud computing.

5 Conclusions

In the article, the most representative classifications of diversity have been analyzed, selected, combined and supplemented. Examples of the construction of various taxonomic structures have been considered. Based on the complexity of describing the types of diversity, it is suggested to use the FIS apparatus for the unification and a corresponding procedure was proposed

The main feature of this classification scheme is that it combines types of diversity from different domains, takes into account new technologies that can be used in safety critical systems. In addition, it is possible to choose ways to solve the security problem by using the resulting classification. For this is necessary to select and apply diversity taking into account the characteristics of domains. An example of using classification for cloud computing was considered.

Further, the diversity classification for IoT, cloud computing, AI, web-services and other modern technologies will be detailed. To do this, it is necessary to design certain filters that will allow moving from a general classification to an individual case of diversity applied to a particular system.

References

1. Preckshot, G.: NUREG/CR6303, Method for performing diversity and defense-in-depth analyses of reactor protection systems. Division of Reactor Controls and Human Factors, Office of Nuclear Reactor Regulation, U.S. Nuclear Regulatory Commission, Washington, DC (1994)
2. Wood, R.: NUREG/CR7007, Diversity strategies for nuclear power instrumentation and control systems. U.S. Nuclear Regulatory Commission, Office of Nuclear Reactor Regulation, Washington, DC (2009)
3. Yastrebenetsky, M., Kharchenko, V.: Nuclear Power Plant Instrumentation and Control Systems for Safety and Security. IGI Global, USA. (2014)
4. Baudry, B., Monperrus, M.: The multiple facets of software diversity: Recent developments in year 2000 and beyond. CoRR. abs/1409.7324, (2014)
5. Larsen, P., Homescu, A., Brunthaler, S., Franz, M.: SoK: Automated software diversity. IEEE Symposium on Security and Privacy. pp. 276-291 (2014)
6. Pullum, L.: Software Fault Tolerance Techniques and Implementation. Artech House, Norwood (2001)
7. Siora, A., Sklyar, V., Kharchenko, V.: (n, m)-version systems: Taxonomy, Models and Technologies. Mathematical Modelling, Information Technologies and Control Systems. 10, 231-246 (2008)
8. Duzhyi, V., Kharchenko, V., Panarin, A., Rusin, D.: Diversity metric evaluation considering extended NUREG-7007 diversity classification. IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT). pp. 21-25 (2018)
9. Schaefer, I., Rabiser, R., Clarke, D., Bettini, L., Benavides, D., Botterweck, G., Pathak, A., Trujillo, S., Villela, K.: Software diversity: state of the art and perspectives. International Journal on Software Tools for Technology Transfer. 14, 477-495 (2012)
10. Kharchenko, V., Gordieiev, O., Fedoseeva, A.: Profiling of Software Requirements for the Pharmaceutical Enterprise Manufacturing Execution System. In: Bris, R., Majernik, J., Pancerz, K. and Zaitseva, E. (ed.) Applications of Computational Intelligence in Biomedical Technology. Studies in Computational Intelligence. pp. 67-92. Springer, Cham (2016)
11. Just, J., Cornwell, M.: Review and analysis of synthetic diversity for breaking monocultures. 11th ACM Conference on Computer and Communications Security 2004. pp. 23-32. Proceedings of the 2004 ACM Workshop on Rapid Malcode, New York (2004)
12. Sen, J.: Security and Privacy Issues in Cloud Computing. In: Ruiz-Martinez, A., Marin-Lopez, R. and Pereniguez-Garcia, F. (ed.) Architectures and Protocols for Secure Information Technology Infrastructures. pp. 1-45. IGI Global, Hershey, PA (2019)
13. Totel, E., Majorczyk, F., Mé, L.: COTS Diversity Based Intrusion Detection and Application to Web Servers. 8th International Symposium, RAID 2005. pp. 43-62. Springer, Berlin, Heidelberg (2005)