UDC 004.415.2. (043.3)

DESCRIPTIVE MODEL OF THE PROGRAMMING STYLE ECOSYSTEM

Sydorov N.A. , Sydorova N.N. , Sydorov E.N.

ДЕСКРИПТИВНАЯ МОДЕЛЬ ЭКОСИСТЕМЫ СТИЛЯ ПРОГРАММИРОВАНИЯ

Sydorov N.A. [a[0000-0002-3794-780X]], *Sydorov N.N*[b[0000-0002-2989-3637]], *Sydorov E.N.*[c[0000-00022609-0230]]

[a]National Technical University of Ukraine, “Igor Sikorsky Kyiv Polytechnic Institute”.

[b]Interregional Academy of Personnel Management, 03039, Kyiv, Ukraine, vul. Frometovskaya, 2.

[c]P&S Integrated Media Enterprise, Avid Development GmbH, Paul-Heyse-Straße, 29, München, Bavaria, 80336.

In the process of developing and maintaining a software product, many things that are called software artefacts are created and used. There is a huge variety of software artifacts, including project plans, work products (specifications, architectural and detailed projects, code, and documentation), user stories, bug reports, tools. Software artifacts can be different in form and presentation. They can be part of a software product or provide processes for its development and maintenance, be intermediate results of processes, or be a part of other software artifacts. Software artifacts are changed, reused, as well as change relationships in the development and maintenance processes of a software product. Therefore, software artifacts play an important role in the software life cycle, regardless of its model, and require attention of all interested parties. The complexity and variety of software artifact relationships require adequate means of description and management. A mean could be a software artifacts ecosystem. In such an ecosystem, a more detailed level than in the software ecosystem is considered. Nevertheless, at this level, the most of the approaches, methods and tools that are used in the software ecosystem can be used. In the article, a concept of software artifact ecosystem is proposed, and a generalized model of the software artifacts ecosystem is presented. The software artifacts ecosystem is the Cornerstone ecosystem type and consists of three types of actors — the platform, the software, and the artifact. The roles of actors in the ecosystem are indicated, the relationships between actors are described. The types, rules, and attributes of actors, relationships, and actions can be specified for models of specific software attribute ecosystems. The same applies to the requirements for analyzing ecosystem properties. A declarative model of the programming style ecosystem based on the generalized model of the software artifact ecosystem, has been developed. The programming style ensuring the comprehensibility of the source code is an important software artifact in the context of the software life cycle. The application of the programming style requires special attention not only from the developers of the software, but also from the project management. This leads to the need to solve the tasks of choosing or developing a style, its application, analysis of the effectiveness of the use of style and change. These tasks can be effectively solved in the context of a programming style ecosystem. In the programming style ecosystem, description of the processes of creation, changing, and use of the programming style is made by applying ontology.Key words: software engineering, software artifact, programming, programming style, software ecosystem, ontology.

1 Introduction

In the development and maintenance of a software product, many things are created and used, which are called artefacts. Artefacts can be different in form and presentation. They can be part of a software product or provide processes for its development and maintenance, be intermediate results of processes, or be part of other artefacts. Thus, there is a huge variety of software artefacts, including, project plans, work products (specifications, architectural and detailed designs, code, documentation), user stories, bug reports, tools, including but not limited to artefact processing. Various and often complex connections are established between artefacts. Artefacts change, are being reused and change connections in the development and maintenance of a software product. Therefore, artefacts play an important role in the life cycle of software regardless of its model and require the attention of all interested parties.

Programming style, while making the source code understandable, is an important artefact in the context of widespread software multiple use and reuse [1]. Applying a programming style requires special attention not only from software developers, but also from project management. This leads to the need to solve the problems of choosing or developing a style, its application, analysis of the effectiveness of the use and changing of style [2].

The software industry is constantly evolving and changing. Not only products and technologies are evolving, but many software companies are experimenting with new business models, which sometimes leads to fundamental changes in all industry structures of both the company and its client. Recently, many companies have been using the concept of "software ecosystems" for development, creating these ecosystems around themselves or their products, taking into account the clients' connections. Ecosystems have shown themselves to be a promising management tool, an evolving software product.

The complexity and variety of software artefact connections require adequate description and management tools. This could be an ecosystem of software artefacts. Such an ecosystem looks at a more detailed level than a software ecosystem, but at this level most of the approaches, methods and tools that are used in a software ecosystem can also be used.

In the article, for the first time, a generic model of the ecosystem of a software artefact is proposed, the application of which is shown on the example of the ecosystem of a programming style. Within the framework of the concept, a generic model of the ecosystem of a software artefact is described, which belongs to the Cornstoun ecosystem type and consists of three actors — platform, software, and artefact. The roles of actors in the ecosystem are indicated, connections between actors are described. Types, rules and attributes of actors, connections and actions can be refined for models of specific ecosystems of the software attribute. The same applies to meeting the requirements for analyzing ecosystem properties.

Based on the generic model of the ecosystem of the software artefact, a declarative model of the ecosystem of the programming style has been developed, which is an artefact that plays an important role in the development and maintenance of software. The description of the processes of creating and using a programming style is made by using the ontology.

2 Related works

Software artefacts. In the life cycle of software, many different auxiliary things (artefacts) are created and used to support the processes of creating and maintaining a software product. Artefacts are created in domain analysis or other lifecycle processes. A wide variety of items are considered as artefacts, from documentation, work products and their parts to supporting tools. By interacting, artefacts enable the efficient execution of lifecycle processes.

In paper [3], artefacts are analyzed in the context of reuse as equipment in the sense of paper [4]. At the same time, three goals are considered (writing, processing and transferring artefacts) and three aspects of equipment (the in-order-to of equipment, readiness-to-hand, presence-and-hand). In addition, since artefacts are analyzed as reusable components that are integrated in the developed software product, their characteristics are taken into account: holism, commonality, reusability and maturity. Considering an artefact as hardware - a thing built into the context of a software product, the mutual influence of the specified characteristics of software artefacts is investigated.

In paper [5], artefacts are considered in the context of a software product line and are divided into three types - architecture, shared components, and components made from shared ones. For each type of artefact, three levels of maturity are identified, depending on the degree of integration of the corresponding type of artefact into the software product line.

In paper [6], artefacts are considered as informational parts that are created, modified and used in the processes of the RUP technology. Artefacts can be of different types and take different forms, from UML models to executable code, and can be used in the development and maintenance of a software product. Artefacts are the input and output of actions in RUP processes. In this case, the main task of the supporting Configuration & Change Management workflow is to manage artefacts created by members of the project development team.

In paper [7], the theoretical foundations of the representation and interpretation of software artefacts are considered. Based on different levels of human perception of artefacts, the user of artefacts introduces three levels of representation of artefacts – physical (physical representation), structural (syntactic structure) and semantic (semantic content). In addition, two steps are introduced for processing artefacts – syntactic analysis of the physical representation (parsing), analysis of the syntactic structure – the result of the first step (interpretation). A meta model of artefacts is built on the basis of presentation levels and processing steps.

In paper [8], software documentation is considered as an artefact, which is defined as a means representing information about software. A model of documentation support as a software artefact is introduced.

The paper [9] considers the architecture of tools that provide the creation and maintenance of metadata about software artefacts that form an environment consisting of resources – development artefacts. Tools are used to manage the artefact environment.

Towards an ecosystem of software artefacts. We are not aware of any paper directly devoted to the consideration of problems associated with the study of ecosystems of software artefacts. However, there are papers, the results of which can be used to solve these problems. In paper [10], attention is rightly drawn to the fact that in software ecosystems, attention is now paid to only the top-level participants – these are organizations and teams that create, implement and maintain software products. However, there is a lower level - artefacts, the role of which in life cycle processes can hardly be overestimated. In paper [11], there are many requirements for describing and analyzing software ecosystems, which are used in our article to model ecosystems of software artefacts.

3 Generic Model of the Software Artefact Ecosystem

This section discusses a generic model of the software artefact ecosystem. Several methods are now used to model software ecosystems [12]. The application of a particular method depends on the type of ecosystem and the goals of the modeling. The i* method [13] is used to represent the ecosystem of the software artefact. Unlike the most commonly used SSN method, which focuses on describing the software ecosystem at the top level (product, developer, supplier, user), the i* method provides an ecosystem description of a more detailed software presentation layer that corresponds to the software artefact level. In fig. 1, method i* presents a generic model of the ecosystem of a software artefact.

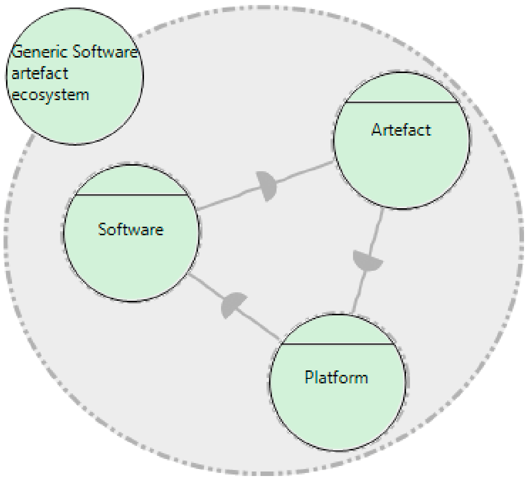

Fig.1 Generic Model of the Software Artefact Ecosystem

When designing an ecosystem, two groups of requirements are met [11]: descriptive and analytical.

The first group includes the requirements for the definition of actors, connections between them and their actions. In addition, requirements are formulated for determining the types, rules and attributes of actors, connections and actions, as well as requirements for determining the specific characteristics of both the ecosystem as a whole and its elements, for example, productivity, efficiency, security.

The second group includes requirements for defining characteristics that provide an analysis of the ecosystem from incentives and motivation to sustainability and productivity.

The table below lists the actors and roles in the software artefact ecosystem. The ecosystem under consideration belongs to the Cornerstone type, since the ecosystem is based on a technological platform for developing and maintaining software, the functionality of which is extended by using artefacts [14]. Thus, the actors of the ecosystem are a platform with an orchestration role, software with a software product role, an artefact with a support service provider role.

Table. Actors and roles in the ecosystem

Ecosystem type	**Actor**	**Role of the actor in the ecosystem**
Cornstoun ecosystem	Platform	Orchestration
	Software	Product
	Artefact	Support service provider

Common connections between actors can be specified. The platform, in the context of which such components as a life cycle model, organizational and technical support for development and maintenance are considered, defines and uses an artefact as an auxiliary means of implementing processes and filling the structure of a software product. Software depends on the platform, which is the main means of implementing development and maintenance processes, and directly, as a component in the software structure, or indirectly, as a means of increasing the efficiency of platform processes, uses an artefact.

Types, rules, and attributes of actors, connections, and actions can be refined for models of specific ecosystems of a software attribute. The same applies to meeting the requirements for the analysis of ecosystem properties [11].

4 Ecosystem of programming style.

To date, methods and tools that are based on multiple and repeated use have become widespread for the development and maintenance of software products. Applying these methods and tools requires the developer to read, analyze, and understand a significant number of representations of work products from different phases of the life cycle. Multiple use and reuse are now commonly deployed from requirements specifications to source code and documentation. Therefore, the requirement of comprehensibility is put forward to the software, one of the main ones. The developer's activities will be more efficient, the software is understandable, and the development and maintenance is cheaper when the styles (standards) are applied when creating the software, ensuring the comprehensibility of the work products of different phases of the life cycle [1].

Fig. 2 shows a model of a programming style ecosystem, which is built on the basis of a generic model of a software artefact ecosystem (Fig. 1).

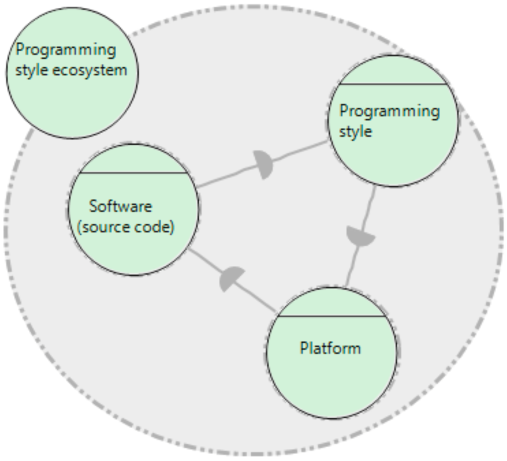

Fig.2 Model of the Software Style Ecosystem

The artefact in this model is the programming style, and the actor, which is the software, is represented by the part of it - the source code for which the programming style is applied. Artefact - the programming style is platform-specific, as the style rules depend on a number of platform conditions, such as the programming language, management goals, timeline, risk, and project budget. In its turn, the programming style is used in the creation of the source code and affects the efficiency of the design and maintenance processes. The use of a programming style involves the implementation of two processes [2]: creation, as a result of which the programming language style is built and the use of the style when writing programs.

In fig. 3. the ontology of creating a programming style is presented, which describes in more detail the participants and actions taking place in this regard in the ecosystem of the programming style. All ontology concepts are categorized as resources in i* terminology, with the exception of the <<event>> concept, which represents a target. At the same time, the concepts Coding phase, Party, Programming language refer to the Platform actor, and the concepts Creating work product style, Style party create guide, Style and Programming language style to the Programming style actor.

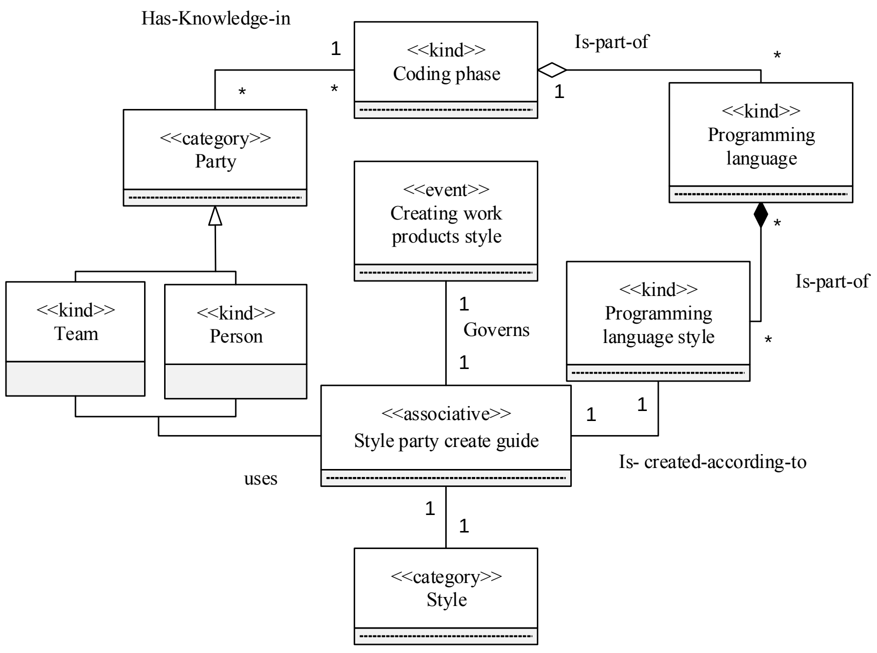

Fig.3. Ontology for creating a programming style

Fig 4. shows an ontology of using a programming style, which also describes the relevant actors and actions in the programming style ecosystem. The Party and Coding phase concepts belong to the Platform actor, the Program, Program style concepts to the Software actor, and the Using work product style, Style party using guide, Program language style concepts to the Programming style actor.

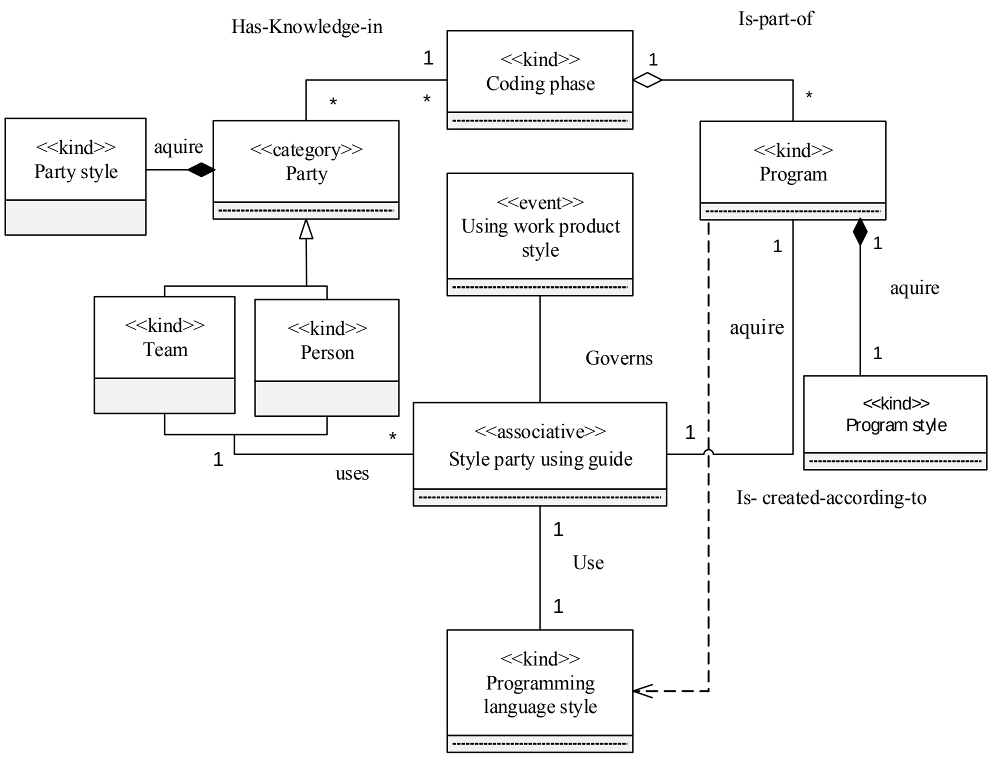

Fig. 4. Ontology of using a programming style

To implement the processes of creating and using a programming style, tools are created that can be considered, on the one hand, as resources of the Programming style artefact, and on the other hand, as artefacts as part of the Platform artefact. These include the programming style knowledge base and the Reasoner (Fig. 5). Thus, the software developer, while coding the program, applies the ontology of the programming style, both for learning the style and for checking the observance of the style in the program. Therefore, two tools are required - one to create an ontology and support the software developer in the coding process, and the second one, to control the application of the programming style in the source code of the program [2].

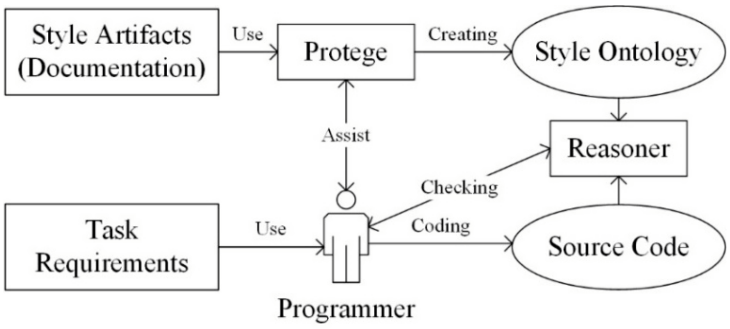

Fig.5 Tools usage scheme

The style analyst, using Protégé, tunes the ontology to the appropriate programming style, creating a TBox (Fig. 6). After setting up, the software developer, using the ontology, is getting used to the programming style with the help of Protégé. The second tool is functionally similar to the Reasoner, but the function of identifying style errors (Style Errors) is added. In terms of descriptive logic, the Reasoner verifies the compatibility of the ontology (Fig. 6).

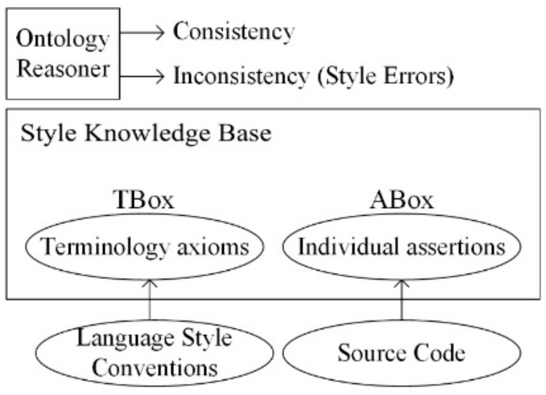

Fig.6. Knowledge base of programming style

Protégé is used to create TBox, part of an ontology, which contains terms that describe a style of programming. Assertions about the source code (ABox) written by the software developer are generated by the corresponding part of the Reasoner, as it provides the corresponding service using the knowledge base (TBox and ABox). The service includes, firstly, the verification of the compatibility of the ontology (a direct function of the Reasoner), and secondly, the search for stylistic errors in the source code of the program.

Summary

Existing approaches to the study of artifacts and tools for their use in the context of the software life cycle are aimed at developing software of a certain type, implementing specific processes or using specific development methods (section 2). Only a unified, systems approach can provide a fundamental basis for the research and use of artifacts in the software life cycle. The purpose of the research in our article is to show that usage of the software ecosystem research methodology, but at a more detailed software artefact level, is productive. An example of a programming style ecosystem case study based on the author's works [1, 2, 12].

For the first time, the concept of a software artefact ecosystem is proposed. Within the framework of the concept, a generic model of the ecosystem of a software artefact is described, which belongs to the Cornstoun ecosystem type and consists of three actors — platform, software, and artefact. The roles of actors in the ecosystem are indicated, connections between actors are described. Types, rules and attributes of actors, connections and actions can be refined for models of specific ecosystems of the software attribute. The same applies to meeting the requirements for analyzing ecosystem properties.

Based on the generic model of the ecosystem of the software artefact, a declarative model of the ecosystem of the programming style has been developed, which is an artefact that plays an important role in the development and maintenance of software. The description of the processes of creating and using a programming style is made by using the ontology.

In continuation of the presentation of the ecosystem, the description of the actors of the ecosystem of the programming style will be expanded and the types, rules and attributes of actors, connections and actions will be developed. In addition, it is proposed to consider the metric provision of the ecosystem in relation to determining the efficiency, sustainability and reliability of the ecosystem of the programming style.

References

1. Sydorov N.A. (2005) Software Stylistics, Problems of Programming. 2018. 2, 3. P. 245–254.
2. Sydorov N., Sydorova N., Sydorov E., Cholyshkina O., Batsurovska I. (2019) Development of the approach to using a style in software engineering, Eastern-European Journal of Enterprise Technologies. 2019. 4/2 (100). P. 41–51.
3. Nuwangi S.M. & Darshana S. (2015) Software artefacts as equipment: a new conception to software development using reusable software artefacts. Thirty-Sixth International Conference on Information Systems, 2015, Texas, USA.
4. Heidegger, M. (1927/1962) Being and Time, Translated by John Macquarrie & Edward Robinson. USA: Harper & Row.
5. Bosch J.,(2002) Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization, Software Product Lines, Second International Conference, SPLC 2, San Diego, CA, USA, August 19-22, 2002.
6. Rational Unified Process: Best Practices for Software development Teams, Rational Software White Paper TP026B, Rev 11/01. 1998. 18 p.
7. Fernandez, D M., Bohm W., Broy M., (2018) Artefacts in Software Engineering: A Fundamental Positioning. International Journal on Software and Systems Modeling. 2018. 26. 9 p.
8. Glass, R. (1989) Software maintenance documentation, SIGDOC '89, Pittsburg, Pennsylvania, USA, ACM Press. P. 18–23.
9. Dewar, R.G., (2005) Managing Software Engineering Artefact Metadata, Department of Computer Science, Heriot-Watt University, Edinburgh, UK.
10. Seichter D., Dhungana D., Pleuss A., Hauptmann B. (2010) Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens, ECSA 2010 August 23–26, 2010, Copenhagen, Denmark. P. 119–126.
11. Sadi M., Yu E. (2015) Designing Software Ecosystems: How Can Modeling Techniques Help?, Springer-Verlag, Berlin Heidelberg 2015. 15 p.
12. Sydorov N. (2010) Software Ecology, Software Engineering. 2010. P. 53–61.

13. Yu E.: (1995) Modelling Strategic Relationships for Business Process Reengineering. Ph.D., thesis. Dept. of Computer Science, University of Toronto. (1995)
14. Knodel J., and Manikas K. (2015) Towards a typification of software ecosystems. In Fernandes et al. Software Business – 6th International Conference, ICSOB 2015, Braga, Portugal, June 10–12, 2015, Proceedings (2015), Vol. 210 of Lecture Notes in Business Information Processing, Springer. P. 60–65.

Получено 17.03.2020

Об авторах:

Сидоров Николай Александрович,
доктор технических наук, профессор.
Количество научных публикаций в общегосударственных базах данных – 18.
Количество научных публикаций в международных базах данных – 2.
https://orcid.org/0000-0002-3794-780X,

Сидорова Ника Николаевна,
кандидат технических наук, доцент.
Количество научных публикаций в общегосударственных базах данных – 15.
Количество научных публикаций в международных базах данных – 7.
https://orcid.org/0000-0002-2989-3637,

Сидоров Евгений Николаевич,
кандидат технических наук, доцент,
старший инженер.
Количество научных публикаций в общегосударственных базах данных – 13.
Количество научных публикаций в международных базах данных – 5.
https://orcid.org/0000-00022609-0230.

Место работы авторов:

National Technical University of Ukraine
“Igor Sikorsky Kyiv Polytechnic Institute”.
E-mail: nyksydorov@gmail.com.
Тел.: 0677980361,

Кафедра компьютерных и информационных технологий
Межрегиональная академия управления персоналом
03039, г. Киев, Украина, ул. Фрометовская, 2.
E-mail: nika.sidorova@gmail.com.
Тел.: 0681006173,

P&S Integrated Media Enterprise
Avid Development GmbH
Paul-Heyse-Straße, 29, München, Bavaria, 80336.
E-mail: Eugen.sidorov@live.com