Method of increasing the reliability of knowledge-oriented systems software through code reuse mechanisms

Метод підвищення надійності програмного забезпечення знаннє-орієнтованих систем за рахунок механізмів повторного використання коду

Serhii Osiievskyi [1 A]
Oleksii Kolomiitsev [2 B]
Pavlo Open'ko * [3 C]
Viacheslav Tretiak [4 A]
Oleksii Petrenko [5 A]
Olha Petrenko [6 D]

[1] Candidate of Technical Science, Associate Professor, Lead Reseacher of the Scientific-research Department of Air Force Scientific Center, e-mail: stiv161272@gmail.com, ORCID: 0000-0003-0861-9417
[2] Dr of Technical Science, Professor, Professor of the Department, e-mail: alexus_k@ukr.net, ORCID: 0000-0001-8228-8404
***Corresponding author**: [3] Candidate of Technical Science, Senior Researcher, Head of the Scientific-research Department, e-mail: pavel.openko@ukr.net, ORCID: 0000-0001-7777-5101
[4] Candidate of Technical Science, Associate Professor, Lead Reseacher of the Scientific-research Department, e-mail: Slava_tr@ukr.net, ORCID: 0000-0003-2599-8834
[5] Candidate of Technical Science, Senior Reseacher, Deputy Chief of the Scientific-research Department, e-mail: alexwgs78@gmail.com, ORCID: 0000-0001-9903-7388
[6] Candidate of Technical Science, Associate Professor, Information Technologies Security, e-mail: petrenkooe73@gmail.com, ORCID: 0000-0002-7862-5399.

[A] Ivan Kozhedub Kharkiv National Air Force University, Kharkiv, Ukraine
[B] The National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine
[C] The National Defense University of Ukraine named after Ivan Cherniakhovskyi, Kyiv, Ukraine
[D] The National University of Radio Electronics, Kyiv, Ukraine

Сергій Осієвський [1 A]
Олексій Коломійцев [2 Б]
Павло Опенько * [3 C]
В'ячеслав Третяк [4 A]
Олексій Петренко [5 A]
Ольга Петренко [6 Д]

[1] кандидат технічних наук, доцент, провідний науковий співробітник науково-дослідного відділу, e-mail: stiv161272@gmail.com, ORCID: 0000-0003-0861-9417
[2] доктор технічних наук, професор, професор кафедри, e-mail: alexus_k@ukr.net, ORCID: 0000-0001-8228-8404
***Corresponding author**: [3] кандидат технічних наук, старший науковий співробітник, завідувач науково-дослідного відділу, e-mail: pavel.openko@ukr.net, ORCID: 0000-0001-7777-5101
[4] кандидат технічних наук, доцент, провідний науковий співробітник науково-дослідного відділу, e-mail: slava_tr@ukr.net, ORCID: 0000-0003-2599-8834
[5] кандидат технічних наук, старший науковий співробітник, заступник начальника науково-дослідного відділу, e-mail: alexwgs78@gmail.com, ORCID: 0000-0001-9903-7388
[6] кандидат технічних наук, доцент кафедри безпеки інформаційних технологій, e-mail: petrenkooe73@gmail.com, ORCID: 0000-0002-7862-5399.

[A] Харківський національний університет Повітряних Сил імені Івана Кожедуба, Харків, Україна
[B] Національний технічний університет "Харківський політехнічний інститут", Харків, Україна
[C] Національний університет оборони України імені Івана Черняховського, Київ, Україна
[D] Національний університет радіоелектроніки, Київ, Україна

Received: February 3, 2022 | Revised: February 22, 2022 | Accepted: February 28, 2022

DOI: 10.33445/sds.2022.12.1.6

Purpose: The problem, that has been considered in the paper, caused by the process of large-scale application of the knowledge-based information systems (KBIS) in critical infrastructure. Research have shown that for KBIS the critical are the errors that can cause harm which significantly exceeds the positive effect of their use. High requirements for the quality and reliability of software for such systems sometimes cannot be adhered to the same due to the real limitations of all types of resources. This has given rise to the occurrence, development and dynamic use of automation methods and tools that ensure the creation of KBIS with the specified high-quality indicators with real limitations on use of the development resources.

Design/Method/Approach: In our opinion, one of the successful ways to achieve the specified quality indicators of software is the development of methods based on code reuse mechanisms. The issues of possibility of application of the code reuse mechanism in the process of designing

Мета роботи: Проблема розглянута у статті, породжена процесом широкомасштабного застосування знання-орієнтованих інформаційних систем (ЗОІС) на об'єктах критичної інфраструктури. Як показали дослідження, для ЗОІС критичними являються помилки, що можуть завдати шкоди, яка значно перевищує позитивний ефект від їх використання. Високі вимоги до якості та надійності програмного забезпечення для подібних систем іноді принципово не можуть бути виконані через реальні обмеження всіх видів ресурсів. Це зумовило появу, розвиток та активне застосування методів та засобів автоматизації, що забезпечують створення ЗОІС із заданими високими показниками якості при реальних обмеженнях на використання ресурсів розробки.

Дизайн/Метод/Підхід дослідження: На нашу думку, одним із успішних шляхів досягнення заданих показників якості програмного забезпечення являється розробка методів в основу яких покладено механізми повторного використання коду. В наведеній публікації розглянуто питання можливості застосування механізму повторного використання коду в and developing software of KBIS have been considered in the paper.

Findings: In frame of the developed method of improving the reliability of knowledge-oriented systems software through the of code reuse mechanisms, a new visual form of function libraries representation in the shape of a single software shell has been proposed.

Theoretical implications: The obtained theoretical provisions are imaged into a running example, which shows one of the possible options for organizing libraries of functions as an element of the information resource.

Practical implications (if applicable the use case diagrams, interaction diagrams, sequence diagrams, class diagrams have been developed and justified.

Originality/Value: Based on the obtained practical results, a block diagram of the method, which, in contrast to existing solutions, includes a procedure for generating of proofreads for basic UML-diagrams in accordance with requirements of the programming environments has been proposed.

Research limitations/Future research: As a basic framework for solving the problem of information resource development, usage of Unified Modeling Language (UML), which is based on the paradigm of object-oriented programming has been proposed. Mentioned choice is justified by the fact that UML is an integral part of a unified software development process and is essentially an open standard that uses graphical notation to create an abstract model of the system.

Paper type: Theoretical.

Key words: UML-diagram, software, source code, information resource, class, function, model.

процесі проектування та розробки програмного забезпечення ЗОІС.

Результати дослідження: В рамках розробленого методу підвищення надійності програмного забезпечення знаннє-орієнтованих систем за рахунок механізмів повторного використання коду запропонована нова візуальна форма подання бібліотек функцій у вигляді єдиної програмної оболонки.

Теоретична цінність дослідження: Отримані теоретичні положення відображені в наскрізному прикладі, що відображає однин з можливих варіантів організації бібліотек функцій як елементу інформаційного ресурсу.

Практична цінність дослідження: Розроблено та обґрунтовано діаграми варіантів використання, взаємодії, послідовності та діаграми класів.

Оригінальність/Цінність дослідження: На основі отриманих практичних результатів запропоновано структурну схему методу, яка, на відміну від існуючих рішень включає процедуру вироблення коректур для основних UML-діаграм за вимогами середовищ програмування.

Обмеження дослідження/Майбутні дослідження: В якості базового інструментарію вирішення завдання розробки інформаційного ресурсу запропоновано використовувати UML (англ. Unified Modeling Language) – уніфіковану мова моделювання, в основу якої покладено парадигму об'єктно-орієнтованого програмування. Зазначений вибір обґрунтований тим, що UML є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення та по суті являється відкритим стандартом, що використовує графічні позначення для створення абстрактної моделі системи.

Тип статті: Теоретична.

Ключові слова: UML-діаграма, програмне забезпечення, вихідний код, інформаційний ресурс, клас, функція, модель.

1. Introduction

The knowledge-based information systems (KBIS) software is characterized by focusing on the subject area for which it is developed and, accordingly, the implementation of complex mathematical models (often unique and inherent in only a certain subject area), which are an algorithmic representation of real physical processes. Quite often, real problems that solvation needs to be implemented in KBIS require using the joint means of several branches of knowledge, which significantly complicates the process of mathematical description and, as a consequence, the software implementation of these tasks. Moreover, a large number of logical-mathematical models are based on algebraic expressions, which requires the use of different methods of calculus mathematics to get the possibilities to obtain numerical calculations with them, which requires professional knowledge of calculus mathematics and is crucial in the software implementation of these mathematical models. Moreover, a significant number of logical-mathematical models are based on algebraic expressions, which requires the use of various methods of computational mathematics to be able to obtain numerical calculations. This point requires the presence of professional knowledge of methods of computational mathematics and is crucial in the software implementation of these mathematical models (Pavlenko, M. A.,2021; Domínguez, Oscar, 2010).

Put in other words KBIS software development consists not only in software implementation of complex application functions that allow to implement system functionality, but also in development of numerous software mechanisms, which assure reliability and correctness of computing process, user-friendly interface, rational usage of computing resources, interoperability of different types of programs within a single project frame. In addition, the software implementation of mathematical models, provides knowledge of relevant applied theories, which related to the specificity of subject areas, by the developers.

The current practice of software development configures with the delimitation of responsibility areas in the process of software development. That means, software developers should be primarily responsible for the implementation of common software mechanisms that ensure the correct behavior of the program within the operating system, as well as the correct interaction of software blocks with each other. At the same time, the task of creating software algorithms that implement the necessary logical and mathematical models is made to a separate task, which should be solved by knowledge engineers, architects and experts. The high complexity and scientific content of this process, as well as the necessity for ensuring the interaction of specialists in different fields of knowledge leads to significant efforts for this process.

That particular circumstance, first of all, shows the necessity for mechanisms for reuse of previously developed and verified program code using as an element of information resources. Put in other words the KBIS development process with usage of current available software design and development approaches is an extremely complex and time-taking process. Despite the fact that on the one hand new approaches to programming allow to significantly increase the efficiency of software development, on the other hand the ever-increasing requirements for functional complexity, program interoperability, ergonomics, etc. of this class of systems require longer periods of their development.

2. Data and methods

Since the occurrence of the first approaches to programming in order to reduce the time consumption for software development, the so-called mechanisms of reusable code have begun to be used and developed (Turinskyi, 2020; Serifi, Veis, 2013; Taylor, Richard, 2007). The essence of these mechanisms is the idea of reusing previously developed software code in new developments.

At the present moment the mechanisms of reusable code include databases or libraries of functions, procedures, classes, objects and agents. To determine a more general category that terminologically combines mentioned above databases and libraries, the paper introduced the term "the information resource".

It should be noted that in the development of general-purpose application software (which does not contain elements of intelligent systems) it is customary to use information-analytical resource that contains theoretical descriptions of tasks with using the points of known methodologies and their practical implementation in known software development environments. Formally, as mentioned above, the information-analytical resource contains databases of functions, agents, class libraries and objects. It is comprehended that object databases contain visual and non-visual components, ActiveX objects, as well as objects of cross-platform implementation. That is, object databases are an assembly of multilevel software mechanisms from making the visualization of properties of instances of the classes in development environments possible to conforming mechanisms for interprogram interaction within both unified and heterogeneous environments. Databases of agents (Vann Tassel, 1995) are a set of implemented agents of different functionality.

Analysis of existing solutions for the development of KBIS showed that using the traditional approach, in which the content of information and analytical resources is an integral part of supporting the development process (Bragina, T. I., 2010; Levykin, V. M., 2013; Osipova, T. F., 2015) is not always appropriate given the diversity of functional needs of participants in the KBIS software development process and needs further research.

3. Research question or Research hypothesis or Problem statement

The objectives of the research will be defined from the perspective of the concept of separate use of analytical and information resources. The issue of forming an analytical resource is elaborated in (Turinskyi, O., 2020; Pavlenko, M. A., 2020; Osiyevs'kyy, S. V.). At the same time the mechanism of information resource formation needs to be researched separately in order to effectively organize the process of KBIS software development at the developers' level.

As a basic tool for solving problem determine above usage of Unified Modeling Language (UML), which is based on the paradigm of object-oriented programming, have been proposed. This choice is justified by the fact that UML is an integral part of the unified software development process and it is essentially an open standard that uses graphical notation to create an abstract model of the system (Maylawati, D. S., 2018; Mouheb D. et al., 2015). In fact, the solution is to implement a mechanism for reusing program (source) code in the KBIS software development process. Namely all applied mathematical expressions and corresponding calculation algorithms are implemented in the form of databases of software functions, classes and objects and establish the core of the information resource for supporting of the process of developing KBIS software at the developer's level.

This will allow:

- create a database of software components for solving typical application problems and ensure the possibility of their reuse in the development of application KBIS software;
- to focus the main efforts of KBIS application software on the implementation of internal software solutions rather than on the implementation of complex application calculations;
- to increase the reliability of the developed KBIS software through the use of already debugged and tested components, and to minimize the possibility of errors due to the "block" nature of KBIS software development.

It is assumed that the creation and widespread usage of databases of software components on the basis of the mechanisms of reusable code in the technological process of application KBIS software developing will provide:

- reliability, unification and standardization of the developed application software;
- increase the efficiency of software development and maintenance by reducing the time spent on direct coding, testing and debugging software.

The structure of the method of improving the reliability of KBIS software through code reuse mechanisms, as a set of relevant models of development, accumulation and usage of software components, as well as a description of the methodology for developing software components databases (as a core information resource) is shown in Fig. 1. This structure gives a generalized idea of the essence of the method and will be detailed due to the developed UML diagrams.

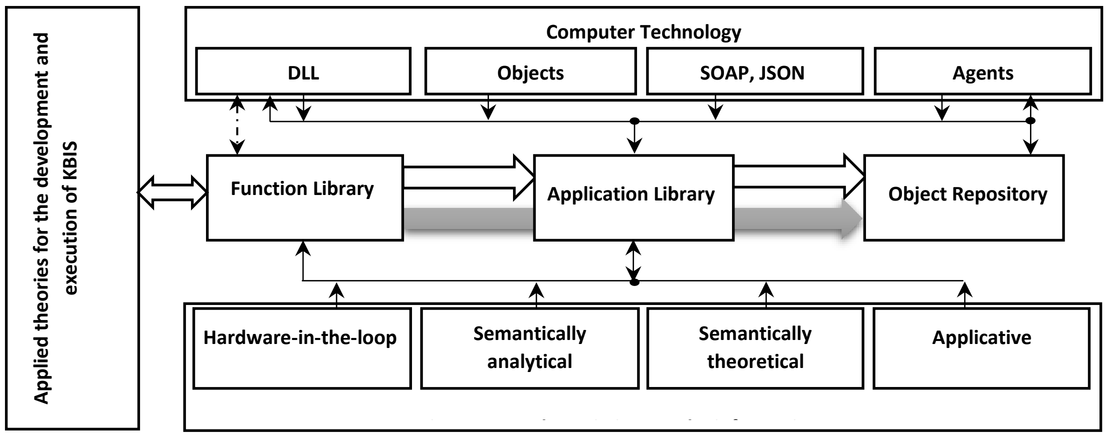

Figure 1 – The structure of the method of improving the reliability of KBIS software through code reuse mechanisms

4. Results and discussion

The very idea of the necessity to develop an information resource containing a database of code reuse software components, and in particular, function libraries for different areas of applied knowledge in the interests of creating different levels software is currently beyond dispute. This is related to the fact that such libraries are the most accessible and at the same time it is a common way to reuse code. This in turns increases the reliability and efficiency of new software development by reducing the time spent on direct coding, testing and debugging software applications (Lipaev, V. V., 2002). First of all, usage of function libraries and other types of databases of code reuse software components increases reliability of newly created software (Levykin, V. M., 2013).

The recent rapid development of computers and no less rapid development of programming methodologies have not reduced the needs of KBIS software developers in function libraries as one of the main mechanisms for code reuse.

This requirement is most often offset by the development of software that includes the most commonly used libraries, such as mathematical, statistical, string, and other functions.

However, the need for directly applied functions, i.e., functions that solve the problems of applied areas of knowledge, which required for the implementation of KBIS software (for instance, such as radar-location, aerodynamics, etc.), still remains unsatisfied. The complexity of the situation is that the existing standards for software development are not customary to be used in the development of function libraries, because of function libraries do not associate as the final product. They are used within the means of development, and their separate standardization has not yet been fundamental. On the other hand, the usage of a common approach to the development of libraries of functions that implement complex mathematical dependencies of applied fields of knowledge may not always be justified. It is related to the fact of sufficient complexity and uniqueness of the fields of knowledge. Moreover, the textual description of such functions used in the documentation for libraries may not always be complete and related to the personal vision of the developer.

That is, source listings and documentation cannot be informative and complete descriptions for application function libraries. There is a need for a new form of presentation of function libraries arises, with taking into account the capabilities of modern programming languages, current requirements and the needs of developers. This issue is especially critical in the distribution of information and analytical resources to support the process of developing KBIS.

In addition, the most applied functions, as off-the-shelf components of program code, are inherent in a contradiction, which is contained in not enough sufficiently clear-cut logical and mathematical description of the domain of function on the one hand, and on the other hand explicit description of the domain of function in software implementation. This contradiction causes a significant number of errors in the synthesis of software code for KBIS software based on software component databases. This situation is usually got worse with the fact that it is impossible to require in-depth knowledge in applied theories in the interests of which software is created from software developers. In addition to the above, the existing function libraries also have internal disadvantages, namely the lack of computer-assisted mechanisms for testing functions, lack of one-way presentation of functions, lack of convenient search mechanisms.

For the foregoing reasons, in the framework of the developed method of improving the KBIS software reliability through the code reuse mechanisms, a new visual presentation form of function libraries by way of a single software shell has proposed. This will allow:

- display a list of all functions implemented in the library;
- display mathematical description and list of arguments for each function;
- carry out the calculation of any of the functions implemented in the library (both with default values and with any values entered by the user) directly in the software shell;
- create a text description of functions in Windows Help format;
- create a listing function with comments in two or more programming languages at the developer's choice.

Such a visual representation of application functions libraries is able to completely satisfy the needs of project developers in the creation and maintenance of software for KBIS.

As was mentioned above, within the framework of the proposed method of increasing the reliability of KBIS software through code reuse mechanisms there will be object-oriented analysis and design of software components, such as visual libraries of application functions, is carried out with using visual tools of Unified Modeling Language UML. A distinctive feature of this approach is the creation of visual models of the developed software application in UML in the process of analysis and design, which is characterized by high semantic saturation. Before starting to create any software, developers must be completely familiarized with the customers’ requirements for the created system. Over extended periods in the process of both object-oriented and traditional software development typical scripts have been used, that allow developers to be more aware of the requirements for the system. However, these scripts were usually interpreted informally, because they were almost always used, but quite rarely documented. This situation was changed in the Objectory software-technological approach (Maylawati, D. S., 2018; Mouheb D. et al., 2015). Such importance has given to scripts (use cases), that they became one of the main elements of analysis and planning of software development project. That is, the use case is a description of a typical user interaction with a computer system. The notation used in UML for describing the requirements for the system makes putting to use of use cases convenient and with sufficient detail level. There is a diagram of use cases for using the function library in Fig. 2.

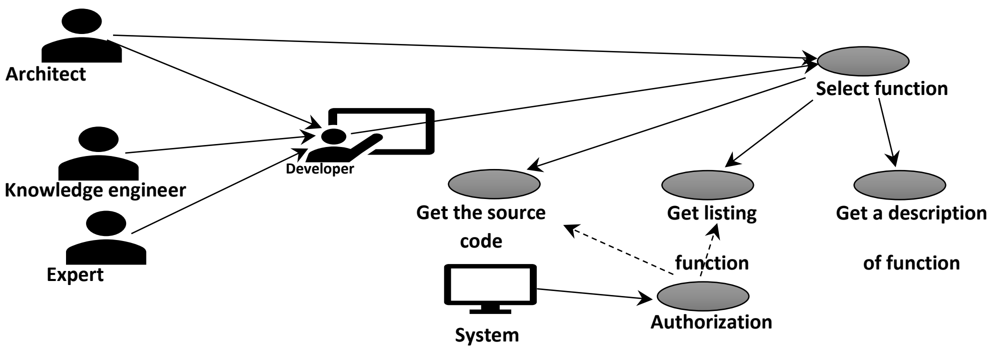

Figure 2 – UML diagram of function library usage options

When use cases diagrams are created an increased focus should be put on implementing and solving specific developers’ tasks. The only limitation is running of external information resources. In this case, the use cases diagrams should show not the user’s tasks but the system interactions that reflect the processes (how the system should perform the designated for user tasks). Each use case is supplemented by a textual description – an event flow. The event flow in standard form describes the events with their intercoupling caused by the implementation of user’s tasks. Nominally given description is the first step in the method of improving the reliability of KBIS software through code reuse mechanisms.

Description of the flow model in the UML notation for each use case is performed on the second stage. Standard format of the event flow model description for one of the use cases, namely the use flow event model of the “Get Function Listing” use case, as an example has shown in Fig.3.

Brief description of this model is as follows. The model allows the user to select the needed function by its name from the list of available options. For the selected function, the main event flow is as follows: the use case starts when the user selects the name of a specific function from the proposed list. When a function is picked out by user, its listing and argument list are displayed. In this case, an alternative flow is to block the task of obtaining a function listing under condition that authorization failed.

Presentation of the requirements for the base of software components in this form allows not only to take into account the requirements for the KBIS software, but also to develop specific options for their implementation, already at the stages of analysis and design.

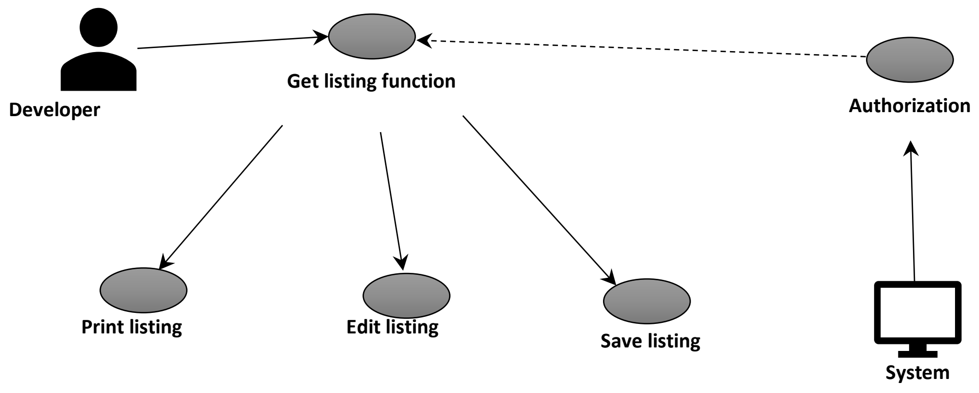

Figure 3 – Diagram of the “Get Function Listing” use case

The diagram of use cases, which given in Fig. 2, displays the highest level of abstraction that contains the project for displaying user tasks. For lower levels of tasks, use cases diagrams are also used and reveal the essence of top-level use cases. Hence, a hierarchical structure of uses of the required degree of detail is created. In particular, in the considered example in Fig. 3 diagram reveals the essence of the “Get Function Listing” use case.

Based on the data from the diagrams of use cases, the elaboration of the main functional blocks of the database of software components for the synthesis of KBIS software is carried out. It means that usage diagrams provide the necessary information for further analysis and design. Such elaboration makes it easy to trace not only the purpose but also the interaction of software blocks. With the view to determine the order of interaction among software modules and their interaction with the system the development of sequence diagrams for each of the main and alternative uses is provided. At its core these are models of program implementation of the project elements interaction.

For instance, for the flow event “Get Function Listing” has been developed a sequence diagram, which is shown in Fig. 4. and the interaction diagram (Fig. 6)

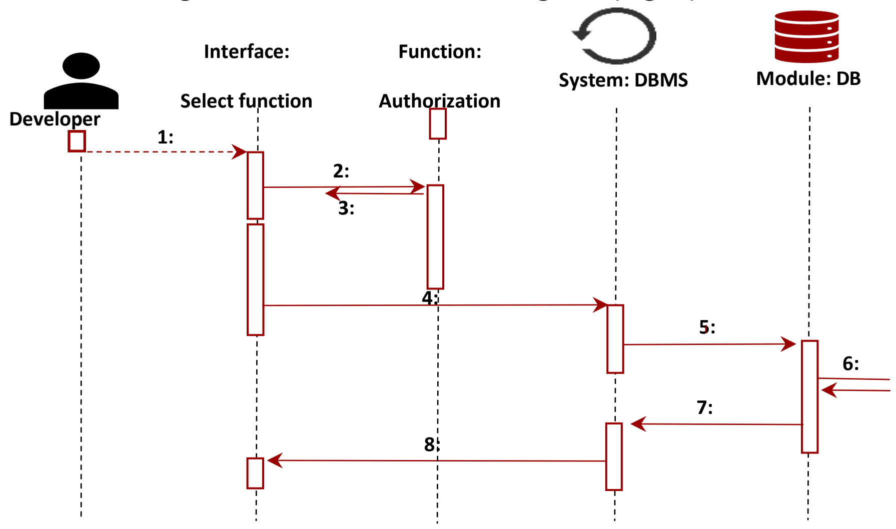

Figure 4 – UML diagram for the flow event “Get Function Listing”

The advantage of using sequence diagrams is the ability to select any acceptable level of abstraction. This is illustrated in Figure 5, which shows a model of the same sequence diagram for the “Get Function Listing” event flow, but at the design phase level of abstraction, which is performed immediately before the start of the software implementation.

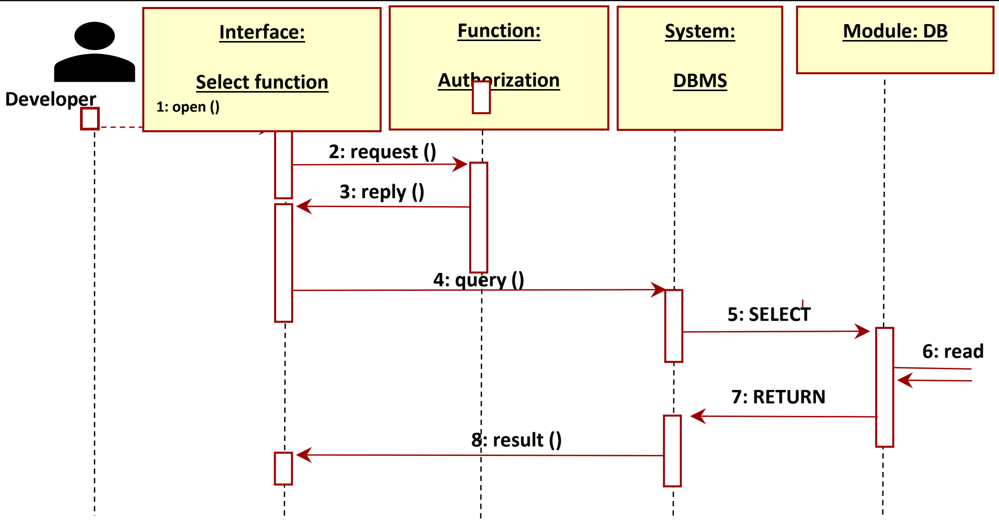

Figure 5 – UML sequence diagram for the “Get Function Listing” event flow with the determining of functions and classes

It should be noted that, as a rule, UML diagrams do not show all actually used or present blocks, classes, objects, connections, entities, etc. Otherwise, this is extremely overloading of diagrams and consequently degrades the quality of their perception. Therefore, the diagrams are used to display only the most important elements, that reveal the essence of the respective software processes.

As a matter of fact, the center link of the all object-oriented methods are class diagrams, as shown in (Mouheb D. et al., 2015; Manuel, Sojer, 2008; Haefliger, S., 2008; Osis J., 2015; Sejans J., 2011). Class diagrams are used to display the types of software implementation objects and the various static relationships that exist among these objects. Class diagrams are able to display the attributes of classes, operations, and constraints that are imposed on coupling between objects as well.

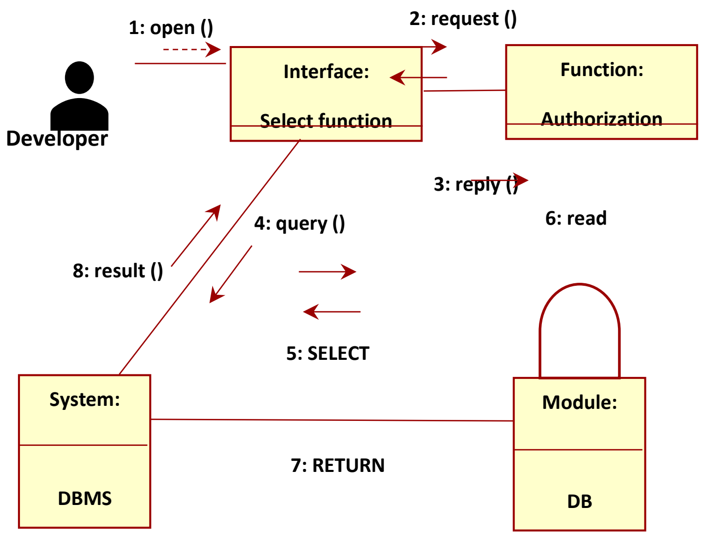

Figure 6 – UML interaction diagram for the “Get Function Listing” use case

Direct design of classes within the framework of the proposed method begins with the highest level of abstraction, where it is convenient to use class diagrams to present a vocabulary of the problem domain. The attributes, operations and connections are determined in the process. In view of the above, the class diagram, which reveals the scheme of the assignment of the main classes of the function libraries, as a typical variant of the database of reusable code software components is shown in Fig. 7.

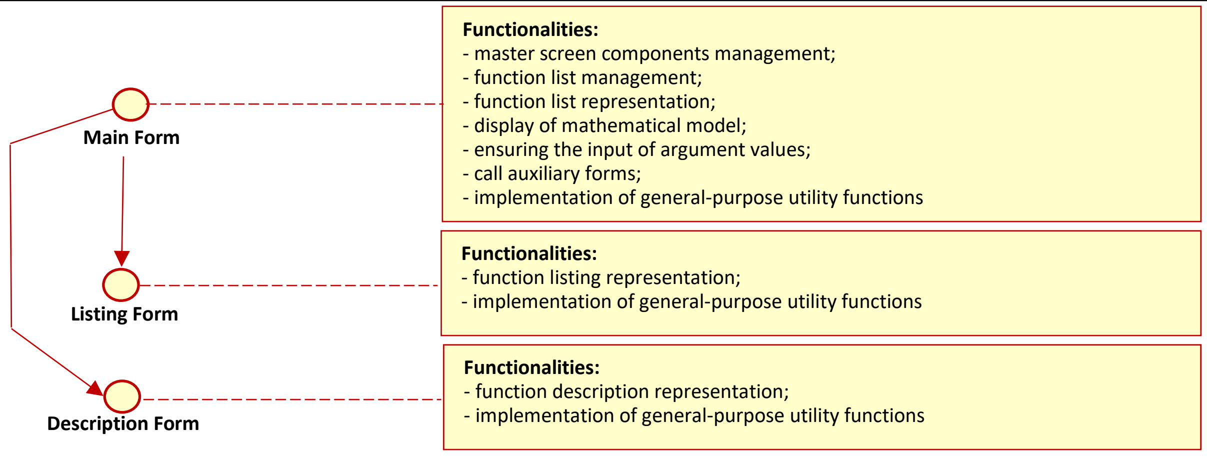
Main Form
Listing Form
Description Form
Functionalities:
- master screen components management;
- function list management;
- function list representation;
- display of mathematical model;
- ensuring the input of argument values;
- call auxiliary forms;
- implementation of general-purpose utility functions
Functionalities:
- function listing representation;
- implementation of general-purpose utility functions
Functionalities:
- function description representation;
- implementation of general-purpose utility functions

Figure 7 – Class diagram of the Main Form of visual representation of function libraries for the highest level of abstraction

Transition to lower levels of abstraction allows to develop a hierarchy of system classes, forming a complete list of classes, that are directly a part of each screen forms with certain names, attributes, operations and couplings. Hence, the class diagram with the addition of lower hierarchy classes will look like in Fig. 8.

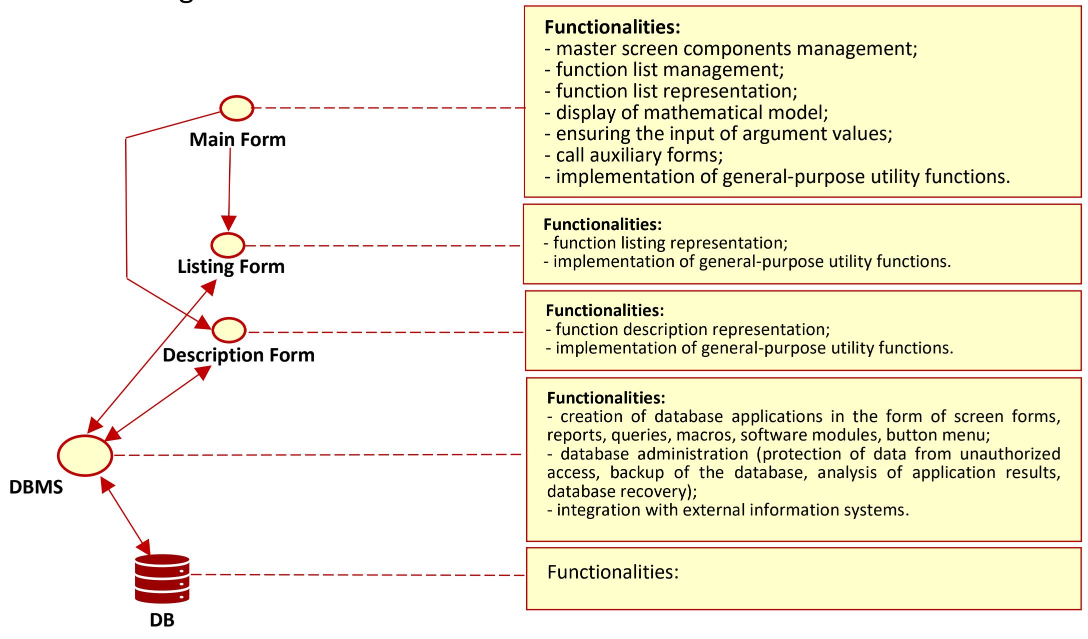
Main Form
Listing Form
Description Form
DBMS
DB
Functionalities:
- master screen components management;
- function list management;
- function list representation;
- display of mathematical model;
- ensuring the input of argument values;
- call auxiliary forms;
- implementation of general-purpose utility functions.
Functionalities:
- function listing representation;
- implementation of general-purpose utility functions.
Functionalities:
- function description representation;
- implementation of general-purpose utility functions.
Functionalities:
- creation of database applications in the form of screen forms, reports, queries, macros, software modules, button menu;
- database administration (protection of data from unauthorized access, backup of the database, analysis of application results, database recovery);
- integration with external information systems.
Functionalities:

Figure 8 – Class diagram of the Main Form of visual representation of function libraries with the inclusion of system classes Control and Table

In this case, the names of the classes are already linked to a specific software implementation, which on the one hand makes it somewhat difficult to read, but on the other hand more convenient in the transition to direct software implementation. Class diagram is a type of static diagram, and it is advisable to use it to explain the static couplings of the projected software components database for the synthesis, development and maintenance of KBIS software.

The most important role in the proposed solution is played the TFunctions class, implemented in the Descript Functions module, and the class diagram of which is shown in Fig. 9.

This module is designed to describe the functions that are directly implemented in the component database of the information resource. Moreover, this module solves the problem of storing the necessary information to represent the implemented application functions.

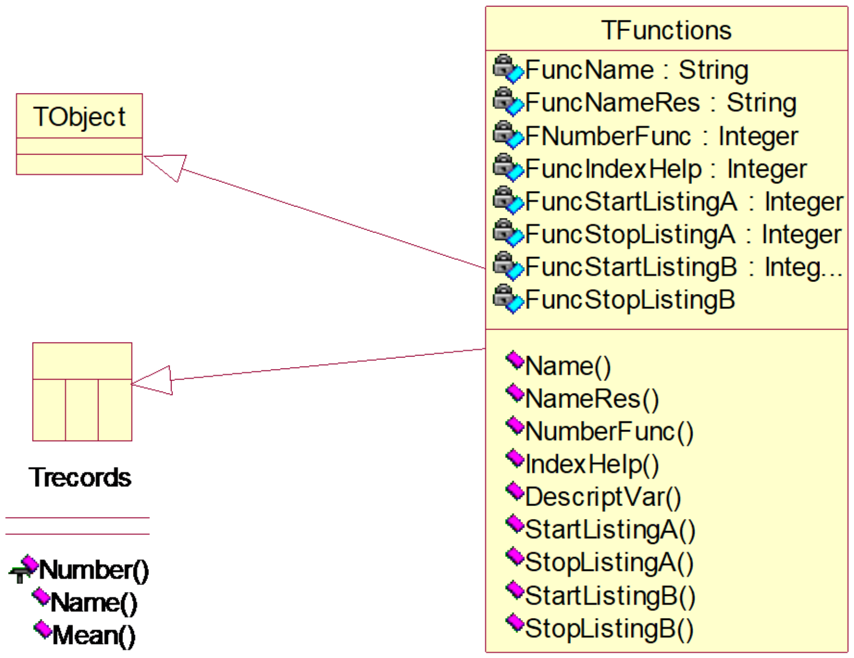

Figure 9 – UML class diagram of the Descript Functions module

Mentioned class is in relation to the association, which directly shows the dependencies among project structures. The TFunctions class diagram shows the fields and properties of this class, each of which shows the visibility type, name, and representation type. As may be inferred from the diagram, according to the object-oriented paradigm, the class fields have got the type of public visibility, and the properties got the type of private visibility. Beside the classes in this diagram, the types of data used in the project are also presented. The final stage in designing a visual library of functions for the synthesis, development and maintenance of KBIS software is the final breakdown of the intended program code into the modules. With this aim in view, within the frame of the proposed method the component diagrams are used.

Hence, the proposed method of increasing the reliability of KBIS software through code reuse mechanisms allows to create deeply verified databases of software components of reused code within the general visual shell. By virtue of certain functionality, the proposed method allows to effectively solve the problem of improving the reliability of mentioned software.

Consistent presentation of the essence of the method in some way eliminates the complex structure (logical sequence) of its implementation. For the avoidance of this fact, a generalized structure of the proposed method of increasing the reliability of KBIS software through mechanisms of code reuse has been presented in Fig. 10.

The proposed method provides not only increasing of reliability, but also reducing span time and increasing the efficiency of KBIS software development on the basis of similar databases of software components (including visual function libraries), as typical representatives of the methodology of developed and verified code reuse.

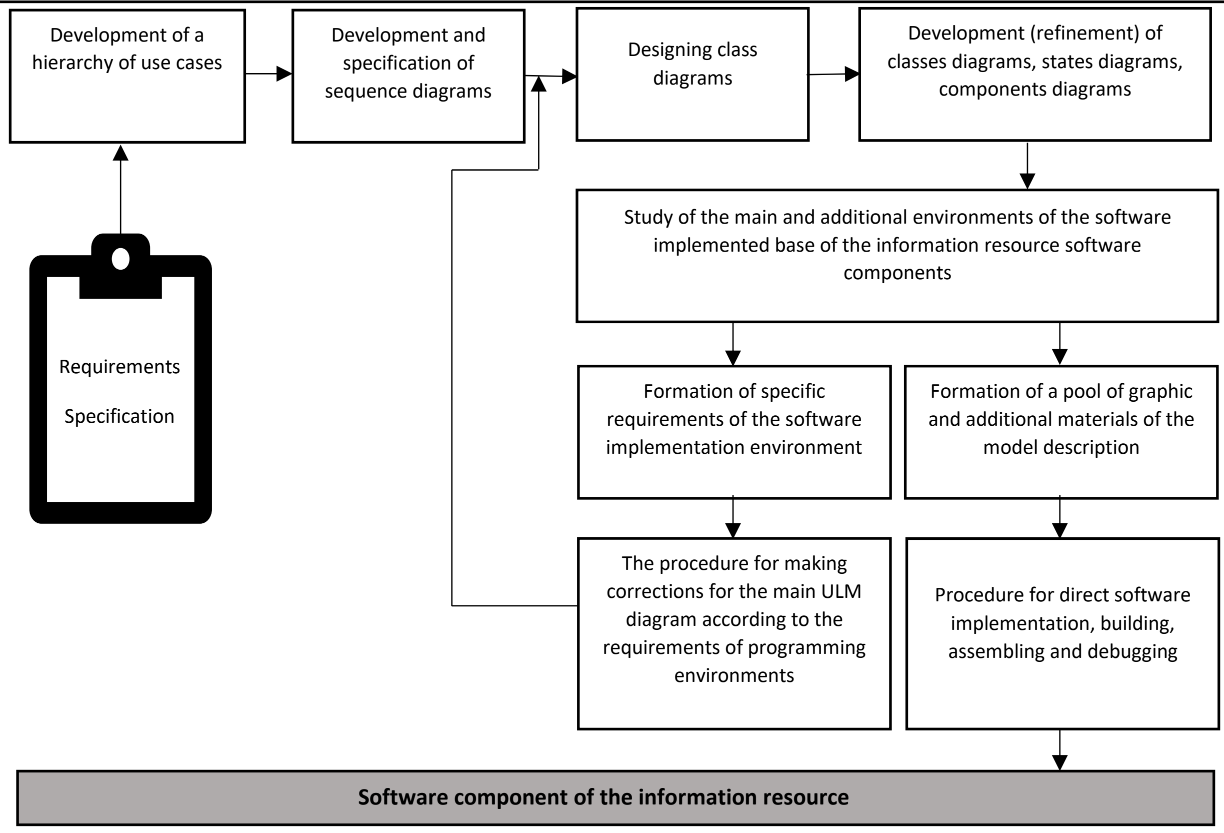
Development of a hierarchy of use cases
Development and specification of sequence diagrams
Designing class diagrams
Development (refinement) of classes diagrams, states diagrams, components diagrams
Requirements Specification
Study of the main and additional environments of the software implemented base of the information resource software components
Formation of specific requirements of the software implementation environment
Formation of a pool of graphic and additional materials of the model description
The procedure for making corrections for the main ULM diagram according to the requirements of programming environments
Procedure for direct software implementation, building, assembling and debugging
Software component of the information resource

Figure 10 – Block diagram of the method of improving the reliability of KBIS software through code reuse mechanisms

5. Conclusions

The proposed method of improving the reliability of KBIS software through code reuse mechanisms has a number of features, both methodological and instrumental, that distinguish it from known and alternative solutions.

In particular, the structural method is designed to ensure the functioning of the system information component to support the development of knowledge-oriented systems. It means that the obtained solutions do not apply to the unstable analytical component. This will allow software developers:

- adequately assess the current quality of KBIS software, in order to develop a corrective effect on design and development flow;
- use the proposed framework at all stages of the life cycle of KBIS software;
- to reduce the level of iterativeness of the technological process of KBIS software development;
- increase the level of complexity of the implemented tasks and, as a consequence, reduce the labour intensity;
- to increase the KBIS accuracy in general.

Reuse of software code, due to the system-defined creation and usage of various databases of software components (libraries of functions, classes, objects and services), provides a significant reduction in the labour intensity of the creating new software process as well as a significant increase in verification and reliability of KBIS as a software product.

The proposed method of improving the reliability of KBIS software through code reuse mechanisms allows to create deeply verified databases of code reuse software components within a common visual shell, which also allows to effectively solve the problem of improving the software reliability.

6. Funding

This study received no specific financial support.

7. Competing interests

The authors declare that they have no competing interests.

References

1. Pavlenko, M. A., Osiyevs'kyy, S. V., Daniuk Y. Methodological Foundation for Improving the Quality of Intelligent Decision-Making System Software. *Information Processing Systems*. 2021. № 1(164). P. 55-64. DOI : 10.30748/soi.2021.164.06.
2. Domínguez, Oscar & Torres, L. M. (2010). Technology intelligence: Methods and capabilities for generation of knowledge and decision making. 1-9.
3. Turinskyi, O., Pievtsov, H., Pavlenko, M., Osievskiy, S., Herasimov, S., Djus, V. (2020). The problem of structuring indicators of quality of decision software support system. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(5), 7916-7923. DOI : 10.30534/ijatcse/2020/144952020.
4. Serifi, Veis & Dašić, Predrag & Ječmenica, R. & D. Labović. (2013). Functional and information modeling of production using IDEF methods. *Strojniski Vestnik*. 55. 131-140.
5. Taylor, Richard & van der Hoek, Andre. (2007). Software Design and Architecture The once and future focus of software engineering. FoSE 2007: *Future of Software Engineering*. 226-243. DOI : 10.1109/FOSE.2007.21.
6. Vann Tassel, D. Style, development, efficiency, debugging and testing of programs [Text]: trans. from English. Moscow: Mir, 1995. 248 p. (In Russian)
7. Bragina, T. I., Tabunshchik, G. V. (2010). Comparative analysis of iterative models of software development. *Radioelectronics, Informatics, Management*. 2010. Issue. 2 (23). P.130–139. (In Russian).
8. Levykin, V. M., Evlanov, M.V. (2013). Model of the architectural framework of accelerated development of the information system. *New technologies*. № 1-2 (39-40). P.51–57.
9. Osipova, T. F. (2015). Modeling of the process of designing an automated information system with the structural method. *Actual*

Список використаних джерел

1. Павленко М. А., Осієвський С. В., Данюк Ю. Методологічні основи підвищення якості програмного забезпечення інтелектуальної системи прийняття рішень. *Системи обробки інформації*. 2021. № 1(164). С. 55-64. DOI : 10.30748/soi.2021.164.06.
2. Domínguez, Oscar & Torres, L.M. (2010). Technology intelligence: Methods and capabilities for generation of knowledge and decision making. 1-9.
3. Turinskyi, O., Pievtsov, H., Pavlenko, M., Osievskiy, S., Herasimov, S., Djus, V. (2020). The problem of structuring indicators of quality of decision software support system. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(5), 7916-7923. DOI : 10.30534/ijatcse/2020/144952020.
4. Serifi, Veis & Dašić, Predrag & Ječmenica, R. & D. Labović. (2013). Functional and information modeling of production using IDEF methods. *Strojniski Vestnik*. 55. 131-140.
5. Taylor, Richard & van der Hoek, Andre. (2007). Software Design and Architecture The once and future focus of software engineering. FoSE 2007: *Future of Software Engineering*. 226-243. DOI : 10.1109/FOSE.2007.21.
6. Ванн Тассел, Д. Стиль (1995). разработка, эффективность, отладка и испытание программ [Текст]: пер. с англ. / Д. Ванн Тассел. – Москва: Мир, 1995. 248 с.
7. Брагина Т. И., Табунщик Г. В. (2010). Сравнительный анализ итеративных моделей разработки программного обеспечения. *Радіоелектроніка, інформатика, управління*. Вип. № 2 (23). С.130–139.
8. Левикін В. М., Євланов М.В. (2013). Модель архітектурної основи прискореного розвитку інформаційної системи. *Нові технології*. № 1-2 (39-40). Р.51–57.

problems of economics and management. № 2(6). P. 89–96.

10. Pavlenko, M. A., Osiyevs'kyy, S. V., Zolotukhina, O. A. Model of support of development processes of intelligent decision-making support systems. 2020. № 4 (69) 130–139 s. DOI : 10.31673/2412- 4338.2020.045051
11. Osiyevs'kyy, S. V., Tretiak, V. F. Model of information-analytical support of knowledge-oriented information systems development processes. Collective monograph: The current state of research in IT-technologies, electronics, engineering, nanotechnology and transport / edited by Goldenblatt M.A. and Valerenko G.I. - Vinnytsia: European Scientific Platform. ISBN: 978-617-7991-47-1, DOI : 10.36074/csriteenat.ed-2.03.
12. Maylawati, D. S., Darmalaksana, W. & Ramdhani, M. A. Systematic Design of Expert System Using Unified Modelling Language, IOP Conf. Ser. Mater. Sci. Eng., vol. 288, no. 1, p. 012047, 2018.
13. Mouheb D. et al. (2015). Unified Modeling Language. In: Aspect-Oriented Security Hardening of UML Design Models. Springer, Cham. DOI : 10.1007/978-3-319-16106-8_2
14. Lipaev, V. V. Software quality guarantee. Methods and standards [Text]. Moscow: Moscow State Technical University "Stankin", 2002. 302 p. (In Russian).
15. Pleskach, V. L., Rogushina, Y. V. Agent technologies: Monograph. Kyiv: Kyiv. nat. trade and economy University, 2005. 344 p.
16. Manuel, Sojer & Joachim, Henkel. 2010. Code Reuse in Open-Source Software Development: Quantitative Evidence, Drivers, and Impediments. SSRN Scholarly Paper ID 1489789. Social Science Research Network, Rochester, NY. Available from : https://papers.ssrn.com/abstract=1489789
17. Haefliger, S., Krogh, G. V. and Spaeth, S. (2008). Code Reuse in Open-Source Software. Management Science, 54(1), pp. 180-193. DOI : 10.1287/mnsc.1070.0748
18. Osis J., Asnina E. Is modeling a treatment for the weakness of software engineering? in: Garcia Diaz V., Cueva Lovelle J., García-Bustelo B. (Eds.), Handbook of Research on Innovations in Systems and Software Engineering, IGI Global, Hershey, NY, 2015, pp. 411-427.

9. Осипова, Т. Ф. (2015). Моделювання процесу проектування автоматизованої інформаційної системи структурним методом. *Актуальні проблеми економіки та менеджменту*. № 2(6). Р. 89–96.
10. Павленко М.А., Осієвський С.В., Золотухіна О.А. Модель підтримки процесів розробки інтелектуальних систем підтримки прийняття рішень. 2020. № 4 (69) 130–139 s. DOI : 10.31673/2412-4338.2020.045051
11. Осієвський С. В., Третяк В. Ф. Модель інформаційно-аналітичного забезпечення процесів розробки інформаційно-орієнтованих інформаційних систем. Колективна монографія: Сучасний стан досліджень в галузі ІТ-технологій, електроніки, інженерії, нанотехнологій та транспорту / за ред. Голденблатт М.А. та Валеренко Г.І. – Вінниця : Європейська наукова платформа. ISBN: 978-617-7991-47-1, DOI : 10.36074/csriteenat.ed-2.03.
12. Maylawati, D. S., Darmalaksana, W. & Ramdhani, M. A. Systematic Design of Expert System Using Unified Modelling Language, IOP Conf. Ser. Mater. Sci. Eng., vol. 288, no. 1, p. 012047, 2018.
13. Mouheb D. et al. (2015). Unified Modeling Language. In: Aspect-Oriented Security Hardening of UML Design Models. Springer, Cham. DOI : 10.1007/978-3-319-16106-8_2
14. Липаев В. В. Гарантия качества программного обеспечения. Методы и стандарты [Текст]. Москва: МГТУ «Станкин»., 2002. 302 р.
15. Плескач В. Л., Рогушина Ю. В. Агентні технології: Монографія. Київ: Київ. нац. торгово-економічний університет, 2005. 344 р.
16. Manuel Sojer and Joachim Henkel. 2010. Code Reuse in Open-Source Software Development: Quantitative Evidence, Drivers, and Impediments. SSRN Scholarly Paper ID 1489789. Social Science Research Network, Rochester, NY. Available from : https://papers.ssrn.com/abstract=1489789
17. Haefliger, S., Krogh, G. V. and Spaeth, S. (2008). Code Reuse in Open-Source Software. Management Science, 54(1), pp. 180-193. DOI : 10.1287/mnsc.1070.0748
18. Osis J., Asnina E. Is modeling a treatment for the weakness of software engineering? in:

19. Sejans J., Nikiforova N. Practical Experiments with Code Generation from the UML ClassDiagram. Proceedings of the 3rd International Workshop on Model-Driven Architectureand Modeling-Driven Software Development, SciTePress, Beijing, China, 2011, pp. 57-67.

Garcia Diaz V., Cueva Lovelle J., García-Bustelo B. (Eds.), Handbook of Research on Innovations in Systems and Software Engineering, IGI Global, Hershey, NY, 2015, pp. 411-427.

19. Sejans J., Nikiforova N. Practical Experiments with Code Generation from the UML ClassDiagram. Proceedings of the 3rd International Workshop on Model-Driven Architectureand Modeling-Driven Software Development, SciTePress, Beijing, China, 2011, pp. 57-67.