22. Kwan, M.-P. Is GIS for Women? Reflections on the critical discourse in the 1990s [Text] / M.-P. Kwan // Gender, Place & Culture. – 2002. – Vol. 9, Issue 3. – P. 271–279. doi: 10.1080/0966369022000003888
23. Pavlovskaya, M. Feminism and Geographic Information Systems: From a Missing Object to a Mapping Subject [Text] / M. Pavlovskaya, K. S. Martin // Geography Compass. – 2007. – Vol. 1, Issue 3. – P. 583–606. doi: 10.1111/j.1749-8198.2007.00028.x
24. Hnatyuk, S. Suchasna veb-kartohrafiya ta yiyi vykorystannya u poperedzhenni y likvidatsiyi naslidkiv nadzvychaynykh sytuatsiy (crisis mapping). Analitychna zapyska [Electronic resource] / S. Hnatyuk // Natsional'nyy instytut stratehichnykh doslidzhen'. – Available at: http://www.niss.gov.ua/articles/806/
25. Kharkiv, Ukraine: Secondary cities [Electronic resource]. – Available at: https://secondarycities.state.gov/kharkiv/#10/49.9804/36.2487
26. Survey123 for ArcGIS: Smarter Forms, Smarter Field Work [Electronic resource]. – Available at: https://survey123.arcgis.com/
27. ArcGISOnline [Electronic resource]. – Available at: https://www.arcgis.com/home/index.html
28. A Guide to the project management body of knowledge (PMBOK Guide) [Text]. – 5-th ed. – USA: Project Management Institute, 2013. – 589 p.

UDC 004.8
DOI: 10.15587/1729-4061.2017.103340

MODELING OF SOFTWARE DEVELOPMENT PROCESS WITH THE MARKOV PROCESSES

T. Savchuk
PhD, Associate Professor*
E-mail: savchtam@gmail.com
N. Pryimak
Postgraduate student*
E-mail: nata.pryimak@gmail.com
*Department of Computer Science
Vinnytsia National Technical University
Khmelnytske highway, 95,
Vinnytsia, Ukraine, 21021

Запропоновано інформаційну модель процесу пошуку і використання асоціативних правил при розробці програмного забезпечення, яка може бути використана при створенні відповідної інформаційної технології. При цьому розглянуто формальні підходи для опису процесу розробки програмного забезпечення. Здійснено моделювання даного процесу на різних рівнях деталізації за допомогою Марковських ланцюгів

Ключові слова: Марковські процеси, Марковські ланцюги, розробка програмного забезпечення, пошук асоціативних правил

Предложена информационная модель процесса поиска и использования ассоциативных правил при разработке программного обеспечения, которая может быть использована при создании соответствующей информационной технологии. При этом рассмотрены формальные подходы для описания процесса разработки программного обеспечения. Осуществлено моделирование данного процесса на разных уровнях детализации с помощью Марковский цепей

Ключевые слова: Марковские процессы, Марковские цепи, разработка программного обеспечения, поиск ассоциативных правил

1. Introduction

Software development is the process of computer programming, documenting, testing, and bug fixing involved in creating and maintaining a program [1]. A software development process is a sequence of stages, the transition between which has no clear boundaries. Usually, the next stage begins upon implementation of 80–90 % of the works of the previous stage. This is especially true of the requirements engineering stage when in some cases evaluation of indeterminate forms occurs only at the end of the project.

In the description of the process of creating software products (SP), the approaches based on data types such as functional, relational (Z, VDM) or axiomatic (OBJ) are preferable. These approaches facilitate software design while being insufficient to describe the system dynamics. Other formal approaches such as finite-state machines [2] or Petri nets [3] allow a detailed description of the system dynamics, but poorly describe changes in internal data during transitions between states. There are approaches that well describe both the system dynamics and processes in data, such as Statecharts [4]. However, they are insufficiently formalized.

The outlined approaches represent the overall software development process in dynamics, but don't represent it at different levels of detail that can be achieved using Markov processes and appropriate mathematical tools.

2. Literature review and problem statement

Formalization of software use cases with the Kripke model has been made [5]. This model is a variation of nondeterministic finite-state machine used in model checking to represent the behavior of a system.

The authors [5] propose to apply a template to transform the description of use cases into a Kripke structure [6]. This structure may further be used for formal system verification [7] or for automatic test case generation [5]. This approach eliminates the gap between informal and formal requirements specification, which will help the users who can't write formal specifications.

This approach would be appropriate for use at the stage of testing data preparation and during software testing [8], but it can't be applied, for example, in software design or programming.

The authors [9] propose to formalize software use cases by means of X-machines similar to the finite-state machines, but have two important differences from them [10]:

– each X-machine corresponds to a certain data set (memory content);

– transitions depend not only on input data, but are also a function of the input value of the data set.

This formalization will provide a complete set of tests for testing a software product. The authors demonstrate the application of the method of transformation of use cases into the appropriate X-machine model on the example of the ATM.

The above formal approach to the transformation of software use cases into different structures should be applied during software testing. However, it does not allow the description of software development process at different levels of detail.

In [11], the authors transform a cognitive map [12] into the Markov model for displaying the development of cognitive project analysis to obtain quantitative estimates of the project state probabilities.

In [11], the authors consider the construction of a cognitive map on the example of software development management. The resulting cognitive map of the software development process represents the system state and transitions between the states. Suppose that the sum of probabilities of all states is unity, and transitions from each state to another are incompatible events. Then such a graph can be presented as a homogeneous Markov chain with discrete states and discrete time [13]. After the completion of a directed graph that represents the cognitive features of software development projects with relations with delays in each of the 10 processes (states), we obtain a Markov chain.

The above-mentioned transformation of a cognitive map into a Markov chain allows passing from qualitative assessments of the software development process to quantitative characteristics. This provides a multi-vector overview of the state of the project development process, but does not represent it at different levels of detail.

The authors [14] suggest designing software using UML diagrams and Petri nets, allowing to find and fix logical errors (looping, end labeling, dead transitions). The creation of software products based on sharing the UML diagrams and Petri nets is divided into several phases: selection and development of appropriate UML diagrams, transformation of the resulting diagrams into Petri nets, Petri nets analysis and making necessary changes in the UML diagram according to the analysis results.

Such transformations are appropriate in software design, but not in considering the software development process at different levels of detail.

The authors [15] solve the problem of predicting the software performance index at the beginning of development. The authors argue that the object-oriented systems modeling language based on components – Palladio Component Model (PCM) allows predicting the software performance. However, the PCM has problems with scalability and provides no correlation between the accuracy of results and overhead analysis. Therefore, the authors suggest using Queueing Petri Nets (QPNs) – an approach to formalization, for which efficient modeling methods based on solution technologies are available.

In [15], the authors present a formal expression of the QPN model based on the PCM, implemented with automated transformation. Experimental data confirm that such an approach provides high accuracy of overhead consideration (up to 20 times lower compared to the PCM approach).

The authors [16] consider the approach to software development – Model Driven Development (MDD), which aims to enhance the role of modeling in software development. The paper deals with the MDD model for the transformation of sequence diagrams into Petri nets.

The authors propose to split a sequence diagram into blocks and represent them by means of Petri nets. Then the blocks can be combined to create a large Petri net. This transformation allows a free choice of Petri nets and eliminates complexity in the analysis of the program developed.

The authors [17] suggest an approach based on the Monte Carlo method for software reliability testing. The outlined approach uses frequent data sets to determine the properties of a given object, and the results represent the percentage of software reliability. This approach is applicable to financial or statistical analysis, software testing, troubleshooting in chains, etc.

The approach based on the Monte Carlo method is useful for software testing, but not for the software development process modeling.

Table 1 shows the comparative description of the above approaches and software development stages suitable for them.

Table 1

Application of formal approaches to software development modeling

Formal approach	Software development stage	Disadvantage
Kripke structure	Test data preparation stage Software testing stage	Impossibility to apply the approach to the description of the software development process at different levels of detail
X-machines	Software testing stage	
Markov chain	Software development management stage	
Petri nets	Software design stage	
Monte Carlo method	Software testing stage	

As seen from Table 1, currently the approaches prevail that describe the software development process only at a certain stage development and cannot represent it at different levels of detail. Thus, there is no approach that allows using a single mathematical tool to describe the process at different levels of detail.

3. The aim and objectives of the study

The aim of the work is to model the software development process using Markov chains.

To achieve this aim, it is necessary to solve the following problems:

– to develop an information model of the software development process;

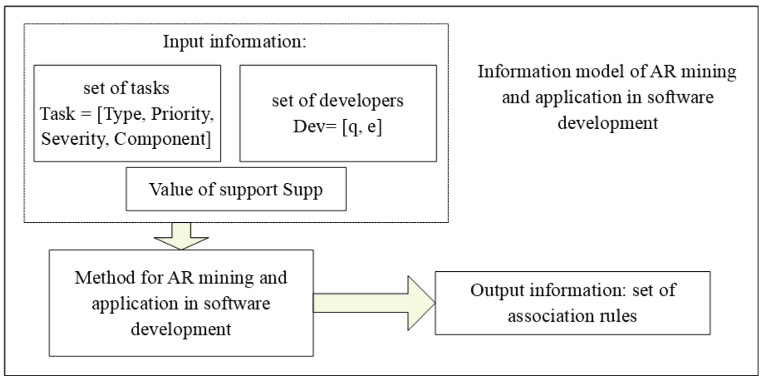
– to define the basic levels of detail of the software development process;

– to define and model the software development stages for each level of detail using the Markov processes.

4. Research methods for the software development process

4. 1. Development of an information model of association rule mining and application in software development

An information model of association rule mining and application in software development (Fig. 1) describes input and output values of the process, as well as parameters and variables present in it.



Input information:
set of tasks
Task = [Type, Priority, Severity, Component]
set of developers
Dev= [q, e]
Value of support Supp
Information model of AR mining and application in software development
Method for AR mining and application in software development
Output information: set of association rules

Fig. 1. Information model of association rule mining and application in software development

Let the i-th task $Task_i$ be described by the following characteristics:

$$Task_i = \langle \text{Type, Priority, Severity, Component} \rangle, \quad (1)$$

where $i=\overline{1,I}$, I is the number of tasks; Type is the type of the i-th task that has the set value {new task, improvement, feature, defect}; Priority is the priority of the i-th task that has a set of values {high, medium, low}; Severity is the degree of importance of the i-th task that has the set value {critical, moderate, minor, cosmetic}; Component is the component of the developed software, the set value of which is dependent on specific software.

Let the j-th developer who performs tasks, Dev_j be characterized by the following indicators:

$$Dev_j = \langle q, e \rangle, \quad (2)$$

where $j=\overline{1,J}$, J is the number of developers; q is the cost of a working hour of the j-developer; e is the experience of the j-developer who has the set value {junior, middle, senior, architector}.

A set of AR found and used in software development is described by:

$$Task_i \cup Dev_j \Rightarrow time_{i_j}, \; Task_i \cup Dev_j \cap time_{i_j} = \varnothing, \quad (3)$$

where $time_{i_j}$ is the duration of the i-th task performance by the j-th developer.

It is necessary to develop an information technology for association rule mining and application, with the quality of the developed software given in specifications, for which the following conditions are true:

$$Time(Task_i, Dev_j) = \sum_{i-1}^{I} \sum_{j=1}^{J} time_{i_j} \to \min, \quad (4)$$

$$Q(Task_i, Dev_j) = \sum_{j=1}^{J} \left(q_j \cdot \sum_{i-1}^{I} \sum_{j=1}^{J} time_{i_j} \right) +$$

$$+ Pr \cdot \sum_{i-1}^{I} \sum_{j=1}^{J} time_{i_j} \to \min, \quad (5)$$

where Time ($Task_i$, Dev_j) is the duration of software development with the quality given in the specifications; Q ($Task_i$, Dev_j) is the cost of software development; Pr is the cost of using technical means per 1 working hour of a software developer (Internet, electricity payments).

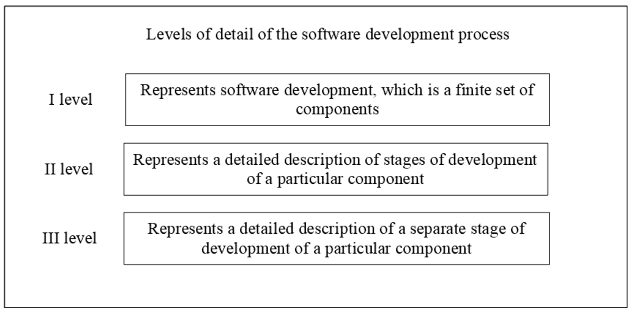
Since this problem can be seen as a multi-criteria optimization problem, two optimization criteria were reduced to one by introducing the efficiency criterion of AR mining and application W as a functional of Time ($Task_i$, Dev_j) and Q ($Task_i$, Dev_j):

$$W = W\big(Time(Task_i, Dev_j), Q(Task_i, Dev_j)\big). \quad (6)$$

The developed information model of the software development process can be used in creating an appropriate information technology.

4. 2. Modeling of the software development process with the Markov processes

The software development process can be seen at the levels of detail (Fig. 2), each being described the corresponding mathematical tool.



Levels of detail of the software development process
I level
Represents software development, which is a finite set of components
II level
Represents a detailed description of stages of development of a particular component
III level
Represents a detailed description of a separate stage of development of a particular component

Fig. 2. Levels of detail of the software development process

The first level of detail of the software development process shows the software product as a finite set of components $product_k$. These components are the programs, being considered as a unit and performing a complete function and used independently or as a part of a software product (SP) (Fig. 3):

$$Product = \{product_1, product_2, ..., product_k, ..., product_K\}, \quad (7)$$

where $k=\overline{1,K}, \; K \in N$.

Each component is developed independently of the other, but the functionality may depend on the performance of other components. Such components can be developed simultaneously and independently by different development teams or in a certain order by one team of programmers. It depends on the project complexity and human resources involved in the development.

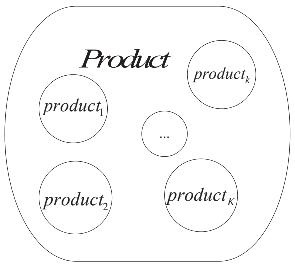


Fig. 3. Graphic representation of the first level of detail of the software development process

The SP development process at this stage can be considered as a random process X(t), whose domain T is a discrete set of points:

$$t_0 < t_1 < ..., \tag{8}$$

and the state space is a discrete set of components Product. At time t_n (where n=0, 1, 2, 3...), one of the components can be developed (i. e., it can be in an appropriate state). At time t_{n+1}, this component can change into a different state or remain the same.

Such presentation of the software development process at the first level of detail corresponds to the mathematical description of the Markov chain – a random process satisfying the Markov property and takes a finite and countable number of states [13]. This process can be described by the conditional uniform distribution function:

$$F_1 = (x_N | x_0,...,x_{N-1}) = F(x_N | x_{N-1}), N = 1,2,... \tag{9}$$

The second level of detail of the SP development process represents a detailed description of the stages of development of a particular component. Each software component is developed according to a specific algorithm. So, the main steps of the algorithm are (Fig. 4).

1. Analysis.

A study of the problem is carried out and the most important requirements to the developed component, from the customer's or users' perspective, are identified [1].

2. Design.

The user's requirements to the component are transformed into detailed and specific requirements to the internal device and its functioning from the programmer's point of view [18].

3. Programming (coding, implementation).

The project is implemented in specific programming languages using specific tools. The result of coding is a finished component of the integrated SP suitable for implementation [18].

4. Testing.

Troubleshooting in the program and documentation is performed and the correspondence between the created component and its specification is determined [19].

5. Documenting.

At this stage, documentation on the finished component from both "external" and "internal" sides is prepared [19].

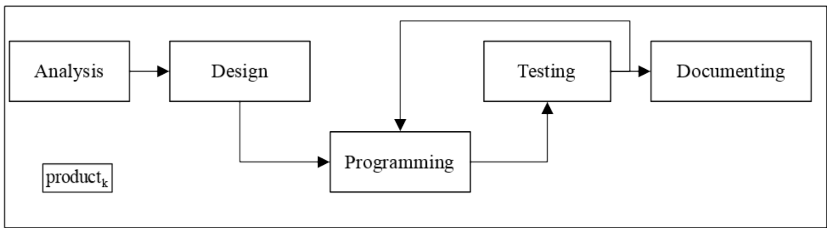


Fig. 4. Graphic representation of stages of the second level of detail of the software development process

At this level of detail, the development process of an individual component can be seen as a random process X(t). The domain T of the process is a continuous set of points t∈T, and the state space S is a discrete set of points $\theta_l \in S$, where $l = \overline{1,L}$. State changes are possible at any random time points $t_0 < t_1 <$.... This process is a discrete random function, for which the one-dimensional distribution function can be represented by [20]:

$$F_1 = (x_N; t_N | x_0,...,x_{N-1}; t_0,...,t_{N-1}) = \\ = F(x_N; t_N | x_{N-1}; t_{N-1}). \tag{10}$$

The third level of detail of the SP development process represents a detailed description of a particular stage of development of a specific component. One of the stages is testing – the process of the program implementation to detect errors or defects [21]. Basic testing steps are (Fig. 5).

1. Preparation for testing [21].

Testing planning includes the actions aimed at identifying key testing purposes and objectives, the implementation of which is necessary to achieve them.

2. Development of tests [21].

Development of tests is a process of writing test cases and conditions based on overall testing purposes.

3. Performance of tests [22].

When performing tests, test cases based on previous test cases are written, a test environment is prepared and tests are started.

4. Evaluation of testing results [21].

After testing, a report that describes the defects found is written and a decision on further bug fixing or changes in the product code is made.

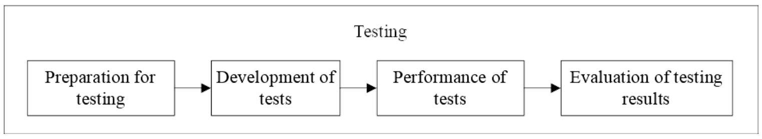


Fig. 5. Graphic representation of the testing stage of the third level of detail of the software development process

Similarly to the first level of detail, the SP development process at this level can be considered as a random process X(t), whose domain T is a discrete set of points

$$t_0 < t_1 < ..., \tag{11}$$

and the state space at this level is a discrete set $S = \{\theta_l, l = \overline{1,L}\}$. State change is possible at time t_n(where n=0, 1, 2, 3...) and such a random process can be viewed as a discrete random sequence of discrete random variables X_N=X (t_N), N=0,1,.... Consequently, this process is a Markov chain [12], whose distribution function is shown in (9).

Thus, each of the levels of detail of the software development process can be modeled using the Markov random processes. So, the first and third levels of detail are modeled using a Markov chain, and the second one – with the discrete Markov process.

5. The results of the research on modeling of the software development process with the Markov processes

The proposed information model based on the Markov processes was used in the development of the Riggoh software product by the following algorithm:

1. The basic components that make up this software product, namely, App, PEW, Notification, Admin, Configuration, GPS, Archive were identified.
2. The basic stages of development of each component were determined. As the development process takes place using a waterfall model, it corresponds to the stages shown in Fig. 4.
3. The basic steps of each stage were determined.

Each of the algorithm steps involved counting of components, stages and steps, respectively.

The research revealed the following patterns:

1. In the analysis of the software development process at the first level of detail, 7 major components that make up the complex Riggoh software product were identified. The number of stages and steps of the development process of the specified software at this level is undefined.
2. The analysis of each of the 7 identified components at the second level of detail revealed 5 basic steps (Fig. 4) required for its implementation. The number of steps of the development process of the specified software at this level is undefined.
3. The analysis of each stage at the third level of detail revealed that the stage of analysis, design and documenting is performed in one step. The programming stage involves the following 3 basic steps: prototyping, test writing, coding. Accordingly, the stage of testing of a software product includes 4 major steps (Fig. 5). Thus, the development of one component of the Riggoh software product at the third level of detail is defined in 10 steps.

Table 2 shows the quantitative indices of modeling of the software development process for each level of detail.

Table 2

The results of modeling of the software development process

Levels of detail	Number of components	Number of stages	Number of steps
I level of detail	7 components	Undefined	Undefined
II level of detail	7 components	5 stages	Undefined
III level of detail	7 components	5 stages	10 steps

Thus, the process of development of the Riggoh software product requires considering ($7\times5\times10=350$) basic implementation steps.

In addition, modeling of the software development process with the Markov chains was performed. The modeling has allowed representing the software development process at different levels of detail, which can be used to develop an appropriate information technology.

At the first level of detail, described by the Markov chain, the number of stages and basic steps of the software development process is undefined, so association rule mining is impossible.

At the second level of detail, described by the discrete Markov process, tasks are not distributed in the software development stages, so, classification algorithms should be used first for their separation. After tasks are classified according to the Type parameter, relationship discovery can be started.

At the third level of detail, defined as the Markov chain, tasks are related to a particular stage of software development, association rule mining can be started immediately using appropriate algorithms [23].

Thus, the use of Markov random processes involves a single mathematical tool to describe a multi-level complex process of software development. Note that a random process can be discrete or continuous at each level. Association relationships between the time needed to solve the software development task and the respective developer would be best to seek for at the second and third levels (Fig. 2).

6. Discussion of the results of the research of the software development process modeling with the Markov processes

As a result of the research, an information model of the software development process was developed. Unlike existing models such as X-machines, Kripke structure, Petri nets and Monte Carlo method, it allows describing the software development process at different levels of detail using a single mathematical tool.

The specified existing approaches to the software development process modeling can be applied to software products based on any principle of software development. While the proposed information model can be applied only to the software product that is based on the waterfall development principle, which is the main disadvantage.

The advantage of the proposed information model is consideration of the software development process at three levels of detail that facilitates understanding.

The research results can be applied in software development for planning this process. Thus, a project manager, knowing the time required to solve a certain task by a developer with specific skills, can effectively distribute all the tasks among team members. This takes into account association relationships on the set terms and quality of the designed software revealed at appropriate stages.

The proposed information model can be the basis for an information technology of association rule mining and application in software development.

7. Conclusions

1. An information model of the software development process that represents the process and can be used in creating an appropriate information technology was developed. A feature of this model is the ability to find relationships between tasks that arise during software development and time necessary to perform them by a certain developer.

2. Three levels of detail of the software development process were identified:

– the first level, representing the development of software, which is a finite set of software components;

– the second level, representing a detailed description of the stages of development of a particular component;

– the third level, representing a detailed description of a certain stage of development of a particular component.

3. Modeling of the development process of the Riggoh software product with the Markov processes at each level of detail was defined and implemented. As a result, the information model of the software development process was obtained.

References

1. Herbsleb, J. D. Global software development [Text] / J. D. Herbsleb, D. Moitra // IEEE Software. – 2001. – Vol. 18, Issue 2. – P. 16–20. doi: 10.1109/52.914732
2. Aho, A. V. The Theory of Parsing, Translation, and Compiling. Vol. 1 [Text] / A. V. Aho, J. D. Ullman. – New Jersy: Prentice Hall, 1972. – P. 147–151.
3. Peterson, J. L. Petri net theory and the modeling of systems [Text] / J. L. Peterson. – New Jersy: Prentice Hall, 1981. – 310 p.
4. Harel, D. Statecharts: a visual formalism for complex systems [Text] / D. Harel // Science of Computer Programming. – 1987. – Vol. 8, Issue 3. – P. 231–274. doi: 10.1016/0167-6423(87)90035-9
5. uz Zaman, Q. Formalizing a Use Case to a Kripke Structure [Text] / Q. uz Zaman, M. A. Sindhu, A. Nadeem // Software Engineering and Applications/ 831: Advances in Power and Energy Systems. – 2015. doi: 10.2316/p.2015.829-017
6. Stirling, C. Modal and temporal logics [Text] / C. Stirling. – GB.: University of Edinburgh, Department of Computer Science, 1991. – P. 23–30.
7. Sindhu, M. Algorithms and Tools for Learning-based Testing of Reactive Systems [Text]: PhD thesis / M. Sindhu. – Stockholm, 2013. – 19 p.
8. Fraser, G. Using model-checkers to generate and analyze property relevant test-cases [Text] / G. Fraser, F. Wotawa // Software Quality Journal. – 2007. – Vol. 16, Issue 2. – P. 161–183. doi: 10.1007/s11219-007-9031-6
9. Dranidis, D. Formal modelling of use cases with X-machines [Text] / D. Dranidis, K. Tigka, P. Kefalas // Proceedings of the 1st South-East European Workshop on Formal Methods, SEEFM'03. – 2003. – P. 72–83.
10. Holcombe, M. X-machines as a basis for dynamic system specification [Text] / M. Holcombe // Software Engineering Journal. – 1988. – Vol. 3, Issue 2. – P. 69. doi: 10.1049/sej.1988.0009
11. Kolesnikova, E. V. Transformatsiia kognitivnyh kart v modeli markovskih protsesov dlya proektov sozdaniia programnogo obespecheniia [Text] / E. V. Kolesnikova, A. A. Negri // Managing the development of complex systems. – 2013. – Issue 15. – P. 30–35.
12. Koshkin, K. V. Kognitivnie modeli upravleniia zhilishchno-komunalnym hozaystvom kak aktivnoy sistemoy [Text] / K. V. Koshkin, S. A. Makeev, G. V. Fomenko // Managing the development of complex systems. – 2011. – Issue 5. – P. 17–19.
13. Tihonov, V. I. Markovskie procesy [Text] / V. I. Tihonov, M. A. Mironov. – Moscow: Soviet radio, 1977. – 488 p.
14. Markov, A. V. Sovokupnoe ispolzovanie setey Petri I UML diagram pri razrabotke programmnogo obespechenia [Text] / A. V. Markov // Sbornik nauchnyh trudov NGTU. – 2011. – Issue 2 (64). – P. 85–94.
15. Meier, P. Automated Transformation of Component-Based Software Architecture Models to Queueing Petri Nets [Text] / P. Meier, S. Kounev, H. Koziolek // 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems. – 2011. doi: 10.1109/mascots.2011.23
16. Jie, T. W. A Model Driven method to represent Free Choice Petri Nets as Sequence Diagram [Text] / T. W. Jie, M. A. Ameedeen // 2015 4th International Conference on Software Engineering and Computer Systems (ICSECS). – 2015. doi: 10.1109/icsecs.2015.7333104
17. Singh, H. Software Reliability Testing using Monte Carlo Methods [Text] / H. Singh, P. Pal // International Journal of Computer Applications. – 2013. – Vol. 69, Issue 4. – P. 41–44. doi: 10.5120/11834-7554
18. Martin, R. Agile Software Development: Principles, Patterns, and Practices [Text] / R. Martin. – New Jersy: Prentice Hall, 2003. – P. 102–103.
19. What are the Software Development Life Cycle (SDLC) phases? [Electronic resource]. – Available at: http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/
20. Gorban, I. Teoriia imovirnostei i matematychna statystyka dlia naukovyh pratsivnykiv ta inzheneriv [Text] / I. Gorban. – Kyiv, 2003. – P. 90–110.
21. Everett, G. D. Software Testing: Testing Across the Entire Software Development Life Cycle [Text] / G. D. Everett. – Wiley-IEEE Computer Society Press, 2007. – 280 p.
22. Fundamentalnii protsess testirovaniia [Electronic resource]. – Available at: http://qalight.com.ua/baza-znaniy/fundamentalnyiy-protsess-testirovaniya/
23. Savchuk, T. O. Poshuk asotsiativnyh pravil dlia pryiniatiia rishen v marketyngovii diyalnosti [Text] / T. O. Savchuk, N. V. Pryymak // Obmin praktychnym dosvidom ta tekhnolohiyamy. – 2015. – Issue 3. – P. 196–199.