https://doi.org/10.31891/csit-2023-2-12
UDC 004.89

Dmytro OKRUSHKO, Antonina KASHTALIAN
Khmelnytskyi National University

SYSTEM OF DISTRIBUTION AND EVALUATION OF TASKS IN THE SOFTWARE DEVELOPMENT PROCESS

The paper is devoted to improving the allocation and evaluation of tasks in software development. Applied aspects of the development of a task allocation and evaluation system are considered in the process of developing software for further analysis, which ensures the most accurate determination of the person who should perform the task and the corresponding task classification tags based on its description. The proposed system provides accurate and fast identification of a person and a group of tags based on the task description. The main goal of the work is to provide an overview of the current state of the art in this field, the advantages and disadvantages of existing approaches, and to propose improvements to the solution.

Challenges related to task allocation and estimation in software development include the need for accurate task estimation, the difficulty of ensuring quality control, and the need for effective communication between developers. To this end, an analysis of the current state of task allocation and estimation was conducted, and a variety of tools and methods available for task allocation and estimation were reviewed, including task tracking systems, project management software, and automated testing tools. Also covered are the various methods used to evaluate tasks, such as peer review, code review, and automated testing.

The future of task allocation and estimation in software development is explored, including the potential for further automation and the need for improved communication between developers, as well as the potential for using artificial intelligence to improve task allocation and estimation. Methods used to measure the efficiency of task allocation and evaluation are also discussed, such as time tracking, percentage of tasks completed, and percentage of defects. The paper proposes AI-based approaches such as natural language processing, machine learning, and deep learning.

Keywords: task planning, task distribution, task classification, frequency characteristics, search for the most optimal criterion for determining the best candidate.

Дмитро ОКРУШКО, Антоніна КАШТАЛЬЯН
Хмельницький національний університет

СИСТЕМА РОЗПОДІЛУ ТА ОЦІНЮВАННЯ ЗАВДАНЬ У ПРОЦЕСІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Стаття присвячена удосконаленню розподілу та оцінювання завдань при розробці програмного забезпечення. Розглянуто прикладні аспекти розробки системи розподілу та оцінювання завдань у процесі розробки програмного забезпечення для подальшого аналізу, яка забезпечує максимально точне визначення особи, яка має виконувати завдання, та відповідних класифікаційних тегів завдання на основі його опису. Запропонована система забезпечує точну та швидку ідентифікацію людини та групи тегів на основі опису завдання. Основна мета роботи - надати огляд поточного стану справ у цій галузі, переваги та недоліки існуючих підходів, а також запропонувати вдосконалення рішення.

Проблеми, пов'язані з розподілом та оцінкою завдань при розробці програмного забезпечення, включають необхідність точної оцінки завдань, складність забезпечення контролю якості та необхідність ефективної комунікації між розробниками. З цією метою було проведено аналіз поточного стану розподілу та оцінки завдань, а також розглянуто різноманітні інструменти та методи, доступні для розподілу та оцінки завдань, включаючи системи відстеження завдань, програмне забезпечення для управління проектами та інструменти автоматизованого тестування. Також були розглянуті різні методи, що використовуються для оцінки завдань, такі як експертна оцінка, перегляд коду та автоматизоване тестування.

Досліджується майбутнє розподілу та оцінки завдань у розробці програмного забезпечення, включаючи потенціал для подальшої автоматизації та необхідність покращення комунікації між розробниками, а також потенціал використання штучного інтелекту для покращення розподілу та оцінки завдань. Також обговорюються методи, що використовуються для вимірювання ефективності розподілу та оцінки завдань, такі як відстеження часу, відсоток виконаних завдань і відсоток дефектів. У статті пропонуються підходи на основі штучного інтелекту, такі як обробка природної мови, машинне навчання та глибоке навчання.

Ключові слова: планування завдань, розподіл завдань, класифікація завдань, частотні характеристики, пошук найоптимальнішого критерію для визначення найкращого кандидата.

Introduction. Problem Setting

Today, developing software for various computer systems is a complex process that requires careful planning, adherence to the plan, and execution. To ensure successful completion of all these tasks, it is very important to have an effective method, a system that will be responsible for distributing these tasks among the performers and evaluating these tasks. After researching some works [1] which presented many tools, information and methods for task distribution and evaluation in the process of software development and computer systems, it was found that the described methods and algorithms have their peculiarities, which do not allow to automate the system of task distribution and classification, so these approaches should be optimized and improved as much as possible, since automation of these processes will affect the coordination in the team, will give the opportunity to determine the exact deadlines for the performance of certain tasks. During the failure of certain nodes of the system, the process of forecasting, distribution and classification of tasks and all elements of the system that were not completed should be transferred to the endpoints or nodes for re-execution. The above-described process greatly affects the speed of the

entire system [2] as a whole and is a bad tone for the end user, although the mechanisms considered often have this concept as a basis, it is possible to improve the mechanisms and methods in different ways.

The purpose of the research. Development of a full-fledged autonomous automated information system for determining the best candidate for task execution, as well as task classification and adding to it pre-defined tags or tag groups.

Object of research. The process of creating an information system with ratings of the best candidates based on their practical skills assessments and classification of the task description using data used in previous learning iterations.

Subject of research. Method and information system for determining the best candidate for task execution and model or method for task classification according to tags.

Research methods. To determine the best candidate for task execution, there should be a method that provides for the creation of a rating system based on ratings given by other users, taking into account the load of a person with existing tasks, as well as taking into account the classification of the task for the presence of tags that a person gives preference to and wants to work with. For determining the classification, there will be a method that provides for the search of a group of tags that belong to a particular task and updating the task structure with new weights for classification tags.

Scientific novelty of results. Creation and modernization of algorithms for creating a new system that will work faster, more accurately and will use fewer machine resources.

Practical significance of the obtained results. Creation of a system with some number of independent APIs that will use cloud technologies and a database to store results, track the accuracy of forecasts and provide fast access to resources and their protection.

1. Analysis of existing tools for task distribution and classification.

Waterfall is a model used during project development. It was one of the first models used in the process of creating various software. The Waterfall model provides developers and managers with a step-by-step plan of work, where before moving on to the next step, all previous ones must be completed, as they are blocking for the next steps. This is very similar to the visualization in Canban, where managers and developer teams can monitor the process of existing and completed tasks with the possibility of monitoring in real time which tasks block the next stage and which tasks are more prioritized. Waterfall is implemented as a model of a sequential life cycle. Waterfall starts with the collection of the team and documentation of all necessary requirements. Then the solution design is carried out, which will solve the set tasks. Then it is all tested, covered with tests and given to the customer. But it is important to note that developers must complete the step before they start a new one.

The Waterfall model has six phases, which are shown in Figure 1. Where each phase was developed in such a way that it was clear when you can start the next step. During the first phase, the team only receives all the requirements from the client, makes basic documentation. They must be thoroughly studied taking into account clearly defined final goals.

On the second phase, the software design team creates all the necessary basic architecture. After determining all the necessary basic structure and necessary software, developers begin the main phase, that is, writing code.

After the basic structure is created, the necessary services are determined and access to third-party services is obtained, developers begin the main phase of implementation of the code, where each module is developed separately, which is then necessarily tested and the entire code is covered with various tests. Also, each module must be fully operational before the next phase.

During the testing and integration phase, all created modules are combined into one system, which will then become the final product. Developers or a special testing team must test the entire system and fix all the errors found.

On the next delivery and deployment phase, developers must install the program or transform it into a state that can be used by end users. Usually during this phase, automatic tests are deployed on servers. And only after the above points, the system falls into the so-called state that is used by end users.

The last phase is the launch of the software, where the developers' task is to support the product or incorporate new customer requests as well as fix any bugs that were not detected during testing.

The waterfall model is very simple to use which gives it a number of advantages such as:

1. Clearly defined phases of development.
2. Offers a large amount of documentation. All stages must be documented.
3. Teams know the amount of work that needs to be done.
4. Presence of clients only at the initial and final stages.

Over time, it became clear that the model also has its drawbacks:

1. Since the requirements may not always be clear, there is not enough flexibility to change the final result in an easy way.
2. Strict division into development phases.
3. These two simple constraints led to failures during development and task distribution, and the reasons were:

4. Presence of unrealistic project goals.
5. Inaccurate project estimates.
6. Presence of risks that are difficult to control.
7. Presence of not precisely defined requirements.
8. Lack of normal reporting on the development status.
9. Lack of communication with the customer, which affects the development process of the project in a negative way and increases the development time when it is necessary to add or correct changes.
10. Usually the use of old or inappropriate technologies for writing code.
11. Also, it may be due to excessive complexity of the project.
12. Presence of commercial pressure.
13. Lack of project development practices.

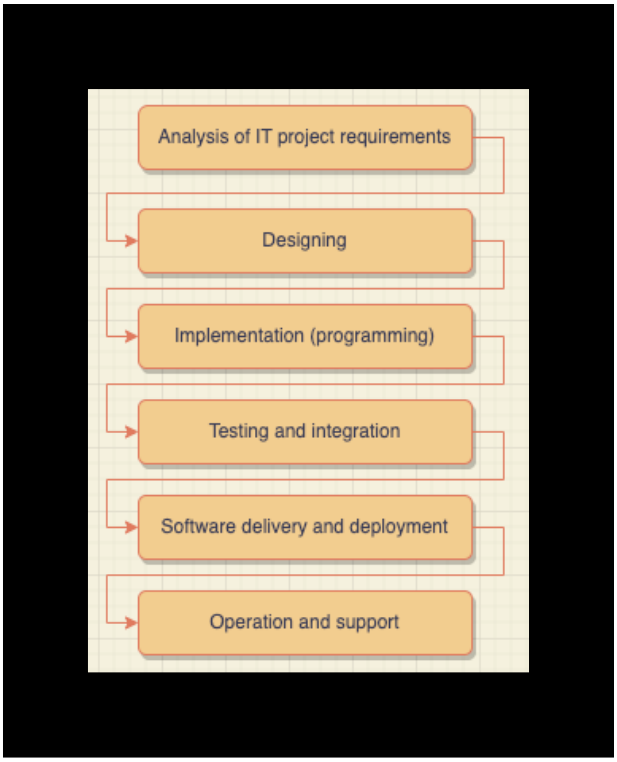

Fig. 1. Waterfall Model

In 2001, the Agile Alliance was formed to create teams of the most flexible and dynamic methods to optimize the software development process. A document was also created that defined the basic concepts of the Agile approach and was called the Agile Software Development (Agile Alliance) [4-7]. Four main concepts were created that defined the basic values of Agile, and these values should be used in all Agile technologies (Figure 2).

The new values and elements contrast sharply compared to the traditional approach, which in turn relied on clearly defined plans. Following the basic principles will help increase the likelihood of success in creating the final product for wide user use.

The differences between the two models that were analyzed and studied are given in Table 1.1.

As practice shows, the total number of failed projects, or projects that are rejected, is quite large. In the work [8, 9] it was reported that only almost 40% of all existing projects are successful. About 45% were rejected for various reasons (there are no all planned functions and capabilities of software, the allocated budget was exceeded, tasks were completed untimely, etc.). And 15% were completely unsuccessful (tasks were completed, but the software did not reach the market, or the order for software was canceled). It was also noted that since 2004 there has been a gradual increase in all successful projects from 30% on average by 5%.

It was also said in the report that the size of the development project has a great influence and this is more important than the chosen methodology regardless of whether it is fast and flexible or traditional.

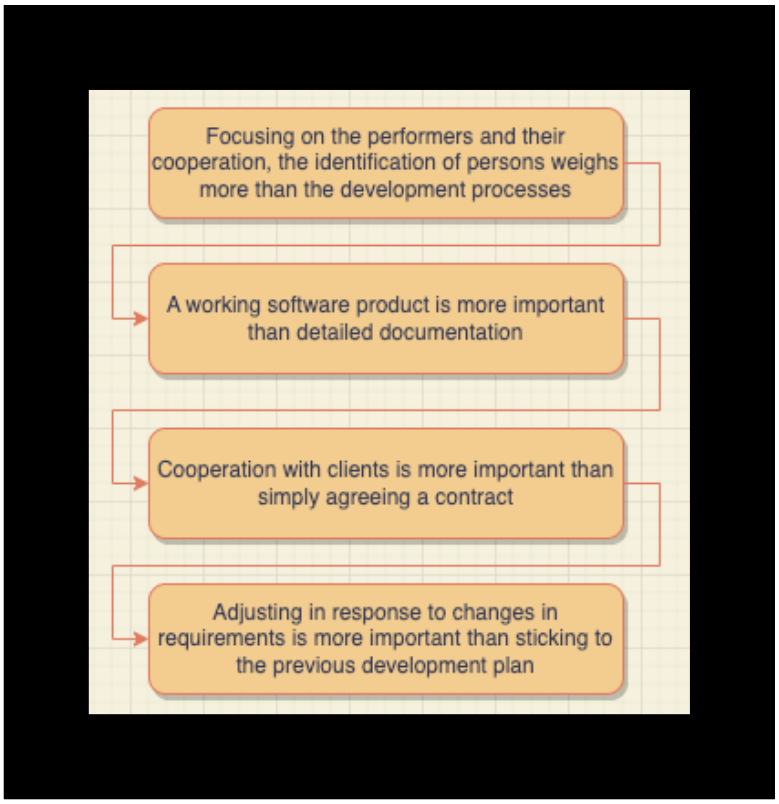

Fig. 2. Four Core Values of Agile Methodology

In the aforementioned work it was stated that a large project is much more likely to encounter problems (10 times) than a small project, and for small projects, flexible methods and approaches greatly simplify the work.

Today, the Agile methodology is expanding in use. As research [10, 11] shows, almost 45% of users who use flexible software development methods use Agile in almost all of their projects. A large percentage of organizations that participated in the survey (90%) say that they use flexible software development methods in all of their projects, as this increases the chances of successful project completion.

The Scrum approach (considered the best method) is used by 55% of Agile teams of managers and developers. If we take into account various modifications of the Scrum methodology, the overall usage rate increases to 73% [12].

Table 1

Comparison of two Agile and Waterfall methods

Agile	Waterfall
Short-term planning with the possibility of dynamic supplementation of necessary requirements.	Planning takes a lot of time since this process requires long-term planning, and requirements cannot be changed during the development of the software according to the approved execution plan and existing infrastructure within the system.
All team members will play an equal role in the team.	All team members follow the established hierarchical structure within the organization.
Constant communication with the client is carried out at all stages of development, which in turn allows to receive timely feedback from the client and understand whether everything was done correctly and clearly.	Completely independent from communication with the customer.
Thanks to the fact that continuous testing is carried out, there is a constant control of the quality of the software.	Late planning can significantly lag behind quality control, as the project is only tested at the last stages of the model and testing takes a lot of time. Also, bug fixes must be structured and present in the system.
It is convenient to measure and track the progress of the product due to measurements in sprints or units.	The entire progress is measured by the steps implemented
The deadlines for software implementation can be conveniently changed if additional requirements for the final product appear during one of the sprints.	Deadlines can be easily determined, since no new changes are expected and if there are no unexpected system failures.
Depending on the projects, they can be divided into different categories such as complex, current or long-term, etc.	The best approach for projects that are not of large size, and all steps can be planned or they are predictable and can be influenced. It is necessary to know all the requirements.

In the IT sphere, there is a widespread belief that teams using Agile in their approach significantly increase the chances of successful software development. Although this is not always confirmed by empirical data, Agile technology has long been seen as a methodology that will make a big breakthrough in software development for projects of various scales. Recent research in [14, 15, 16] studying the success of Agile projects shows that the possibility of such a belief can be proven on scientific data. The works showed a quantitative study to check whether the use of flexible Agile methods during software development affects the success rate of products. They noticed signs that the use of flexible planning methods in the software development process has a significantly higher registered success rate of products. Evaluation and display of results were conducted for three categories of success: overall project success, stakeholder success, and effectiveness.

Agile projects are well known for their flexible software development process compared to traditional projects. The process itself consists of a large set of practices that in turn describes a set of procedures that teams of developers then use to achieve the project's final goals.

For IT projects with the goal of creating quality software, the agile approach helps to determine the scope and size of the project at all stages, including the final stage when the project is in constant use by users. Size, required time, as well as cost and quality are quite important criteria when considering the success of a software product.

2. Development of an algorithm that will allow automation of the task distribution process.

In the begin, it is necessary to form a corresponding list of skills for each team member (Figure 3).

Fig. 3. A list of necessary skills describing a person

The maximum total score of all skills will be 160 points, since there are 16 skills in total. To find out how much each skill is worth in percentage, we divide each skill by 6 and get 6.25%. - . This is the maximum value that a skill can get.

Next step, we will divide these skills into 4 main groups. These are Skill (S), Management(M), Reliability(R), Prospects(P). Let's calculate the sum of all skills that belong to the Skill group.

$$S = \sum \begin{pmatrix} DES(m), UIS(m), BS(m), DOS(m), RNS(m) \\ , TS(m) \end{pmatrix},$$

where m - this is the maximum grade in percentage. Let's now calculate the average and relative value of each skill for this group.

$$S(a) = S/6,\ S(r) = \frac{S}{0{,}375},$$

де 0,375 це –

$$(S_{max} * 6)\ /100.$$

Now let's calculate the corresponding ratings for Management (M), Reliability (R), and Prospects (P) using the same formulas.

$$M = \sum\{E(m), SM(m), TM(m), ED(m), S_a\},$$

Where S_a – this is a previously calculated value, i.e. the average value for Skill, and m is the maximum rating in percentage.

$$R = \sum\{Ca(m), SM(m), TM(m), ED(m), Re(m)\},$$

Where m - this is the maximum grade in percentage.

$$D = \sum\{LS(m), DS(m), S\},$$

where *m* – This is the maximum score in percentage ratio, and S is the sum of all skills of the Skill group. And accordingly, the average values.:

$$M(a) = M/5,$$

$$R(a) = B/4,$$

$$D(a) = D/3,$$

And relative meanings:

$$M(r) = \frac{M(a)}{31{,}25},$$

Where 31,25 is –

$$(S_{max} * 5)/100,$$

$$R(r) = \frac{R(a)}{0.25},$$

where 0,25 is –

$$(S_{max} * 4)/100,$$

$$D(r) = \frac{R(a)}{n},$$

Where n is –

$$D(r)\frac{(S_{max} * 2 + S(r))}{100}.$$

For the tag determination system, a data set needs to be created. This data set is formed from a graph where each vertex is connected to another vertex, i.e., a many-to-many relationship.

We will identify 5 main tag groups. These will be: Priorities, Projects, Skills, Types, Phase. Besides these main groups, there may be others. Each such tag group can contain a subset of size N, where N is a natural number. To determine the tags that are related to the task, we will use the formula:

$$Tags\{N\} = \sum_{i=1}^{N}(\overrightarrow{max}(Priotities\{SubPriorities(i)\}) + \sum_{i=1}^{N}(\overrightarrow{max}(Projects\{SubProjects(i)\}) + \sum_{i=1}^{N}(\overrightarrow{max}(Phases\{SubPhases(i)\}) + \sum_{i=1}^{N}(\overrightarrow{max}(Skills\{SubSkills(i)\}) + \sum_{i=1}^{N}(\overrightarrow{max}(Types\{SubTypes(i)\}),$$

where SubPriorities $\in$ Priorities, Sub Projects $\in$ Projects, SubPhases $\in$ Phases, SubSkills $\in$ Skills.

To determine who will be responsible for completing the task, a rating system needs to be created. The ratings include: the number of tasks completed by tag group, the number of tasks completed by subgroups, the number of tasks planned to be completed, the total sum of estimations for all planned tasks, the total sum of ratings, S(r) value, M(r) value, R(r) value, D(r) value, and the sum of all other skills. After creating the corresponding ratings with positions in the sub-ratings, we form one general rating. Then, taking into account the preferred and not preferred tags of each participant, we lower or raise his position in the rating.

To calculate the impact of preferred and not preferred tags, we use the following formulas:

$$preferred = \prod_i 10 * \{P_i\}/100,$$
$$not\ preferred = \prod_i 10 * \{P_i\}/100,$$

Where - P(i) is a set of preferred and not preferred tags for a task.
The formula for the influence on the overall rating in the ranking is:

$$mark = total\ mark + (preferred - not\ preferred).$$

To influence the general time for task completion, simply multiply the rating by the corresponding value in the time range {2,5,10,15,20}. The next step is to find the maximum value in the rating and in this way we find out who our potential person is best suited for the task.

4. Results of distribution and evaluation of tasks

For the experimental research, a dataset generated by us was used. To generate the dataset, a graph structure was taken as a basis, where each node has a connection with all other nodes (Figure 4). For a simple visualization of the prediction of the optimal candidate, let's take as an example the description of the task - "Me as a super admin, should be able to create a form for inviting users", and no qualification tags will be added. The results of auto-tagging the task can be seen in Figures 5, 6. In Figure 5, it can be seen that the best classification tag for priority is High. Sprint is the best for phase (Figure 6). With a big gap in Skills, the tag is React. And for the mandatory Type we have Task.

Next, the obtained tags need to be used to search for the best candidate to perform the task.

At the first stage, you can see the rating of all players for all tags (Figure 7). Then from the current rating it is necessary to form a new one taking into account the load of people. Then another one with the influence of preferred tags (Figure 8) and not preferred (Figure 9). Then build the last rating to select the best performer of the task (Figure 1.10). From the example we see that Attendee - 8 is the best option.

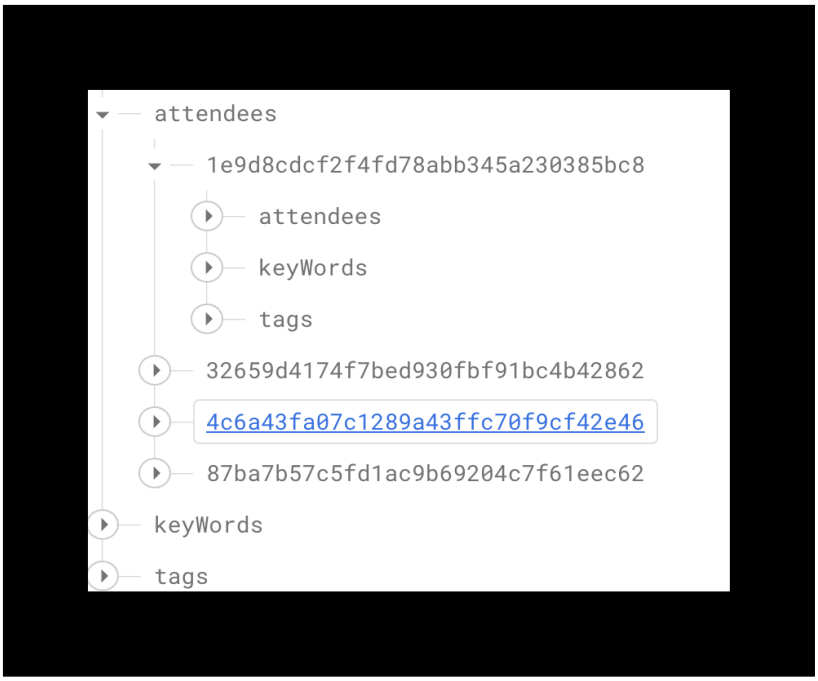

Fig. 4. Dataset structure

In Figure 4 you can see a clearly structured entity where each key has an attendee Id which is hashed. In turn, each attendee has a link to each attendee, including itself. This includes the key words which are also hashed, of the task described and all existing and used tags in the system. This will help in the future when we generate the most compatible and appropriate tags for task description.

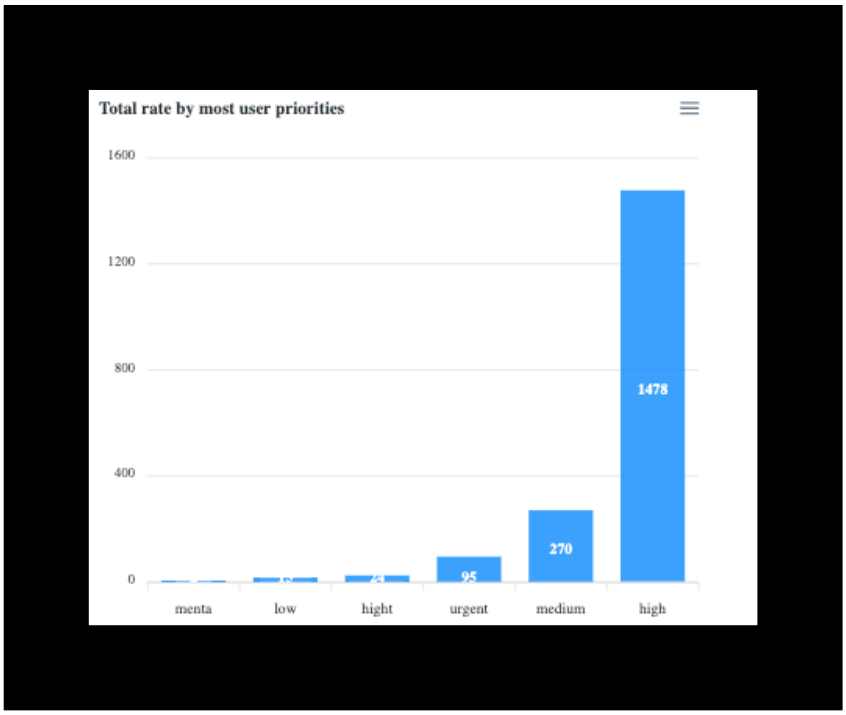

Fig. 5. Total rate by most used priorities

In the Figure 5 it can be seen that out of all the tags present in the graph structure (height, low, urgent, medium), the tag 'high' is the most frequently used for all the words described in the task "Me as a super admin, should be able to create a form for inviting users". This tag was used 1478 times. It has a huge gap from the tag 'medium' which has only 270. This suggests that the tag 'high' is probably the most optimal option for the described task.

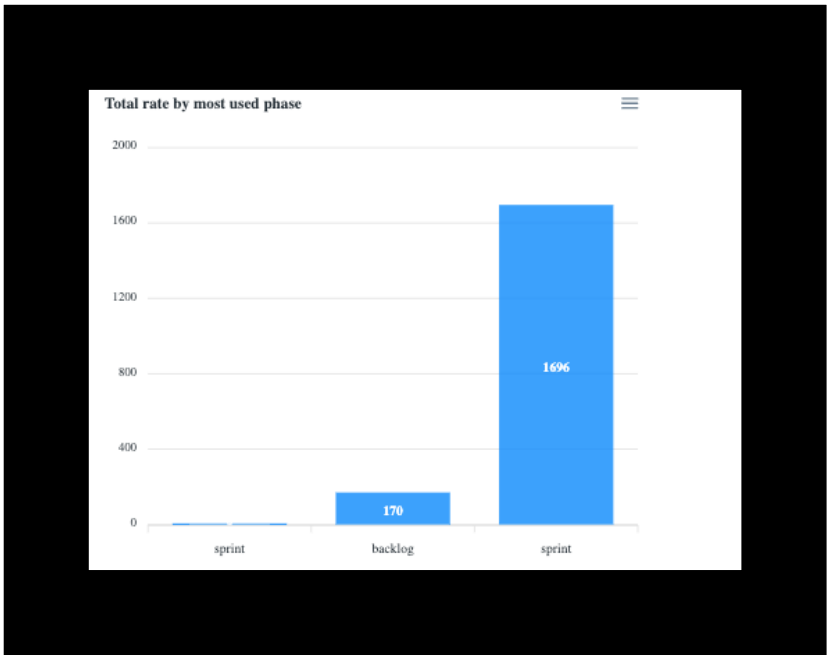

Fig. 6. Total rate by most used phase

In the Figure 6 it can be seen that out of all the tags present in the graph structure (sprint, backlog), the tag 'sprint is the most frequently used for all the words described in the task "Me as a super admin, should be able to create a form for inviting users". This tag was used 1696 times. It has a huge gap from the tag 'backlog' which has only 170. This suggests that the tag 'sprint' is probably the most optimal option for the described task.

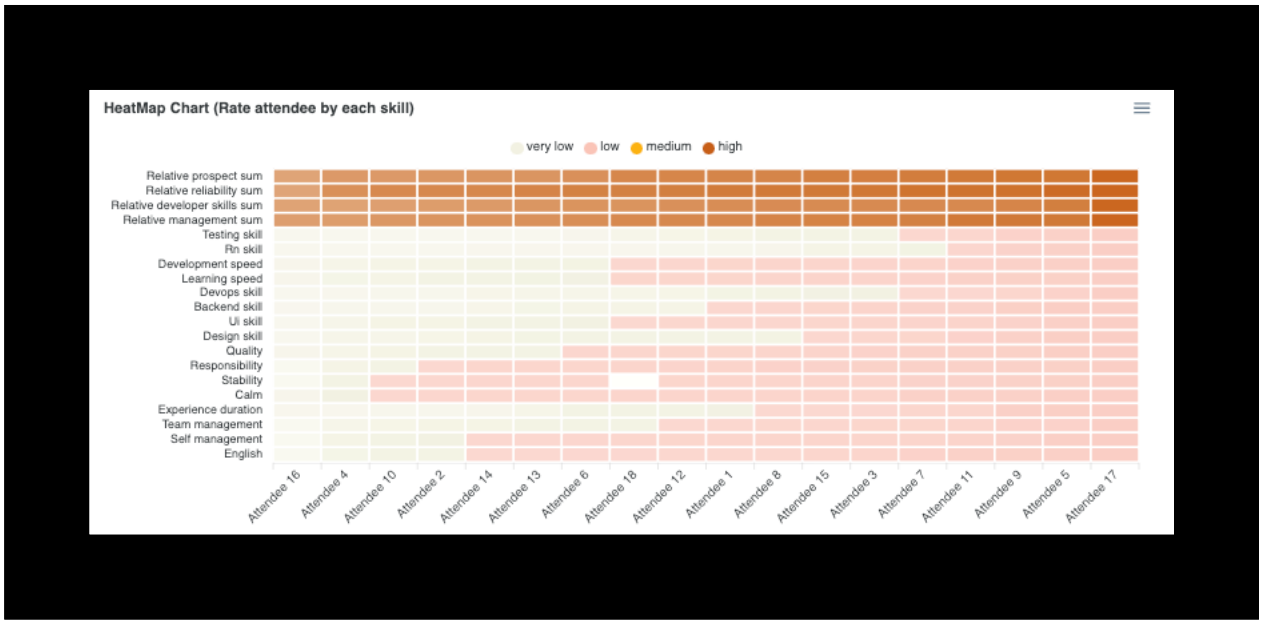

Fig. 7. Total heatmap of attendees by all tags

In the Figure 7, you can see the rating of the attendees and the rating of each attendee for each skill. High is the value from 10 to 110, medium is the value from 7 to 10, low is the value from 3 to 7, and very low is the value from 0 to 3. As we can see from the graph, the top 3 best candidates according to the ratings of all skills are Attendee 9, Attendee 5, and Attendee 17. However, these top 3 are not our most optimal candidates yet, as we need to make a rating list only for the tags we got in Figures (3-4) and also take into account the preferred and not preferred tags of each participant in the rating.

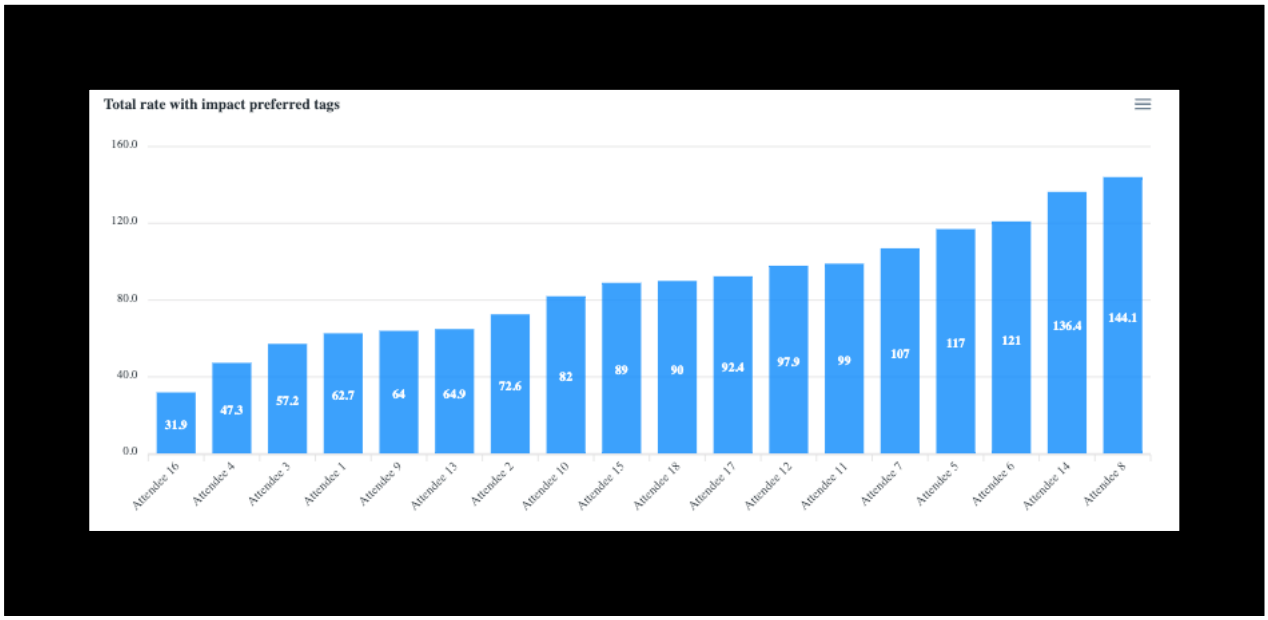

Fig. 8. Total rate with impact preferred tags

Taking into account the tags that participants would not like to work with, we added all the scores by tags and took into account the wishes we received, we got one overall rating as in Figure 6. From it we see that the candidates who were in the previous rating Attendee 9, Attendee 5, and Attendee 17 changed their position in this rating. Now the top 3 are taken by participants Attendee 6 - 121 points, Attendee 14 - 136.4 points, and Attendee 8 - 144.1 points. The next step is also to take into account the tags that participants did not want to work with in the already fading rating.

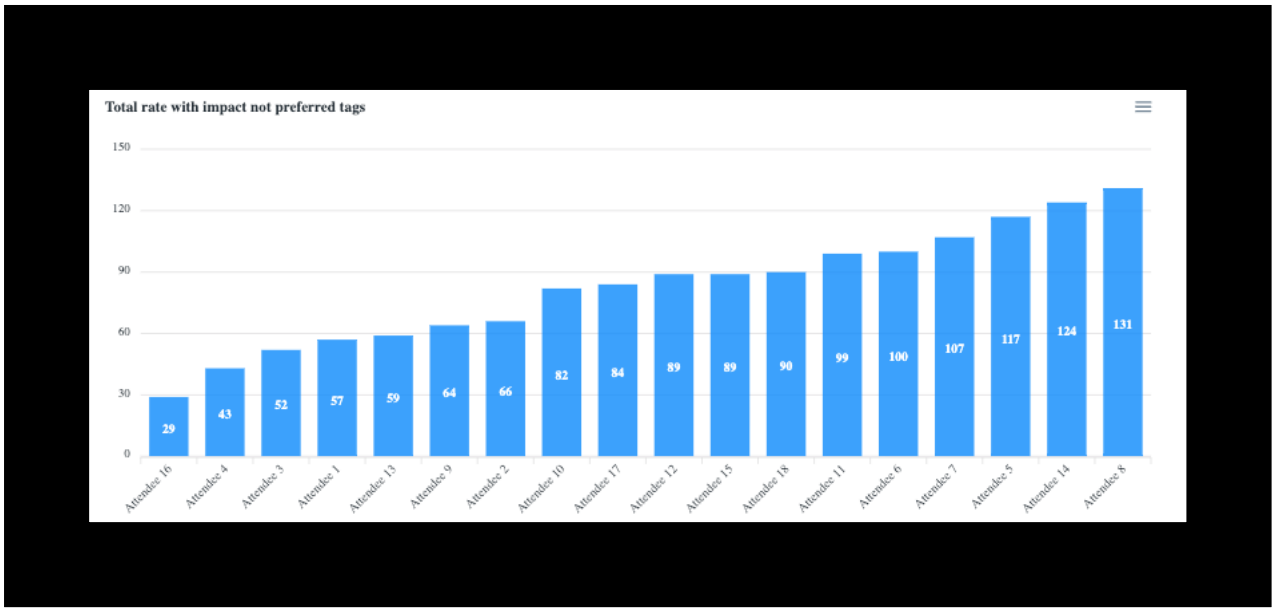

Fig. 9. Total rate with impact not preferred tags

Taking into account the tags that participants would not want to work with, we changed the ranking based on the number of tags that participants would not want to work with. We got one overall ranking as shown in Figure 7. From it we see that the candidates who were in the previous ranking Attendee 6 - 121 points Attendee 14 - 136.4 points, and Attendee 8 - 144.1 points changed their position in this only Attendee 6 to Attendee 5. Now the top 3 are occupied by participants Attendee 5 - 117 points Attendee 14 - 124 points, and Attendee 8 - 131 points. As we see, the scores of all participants from the top 3 have decreased. This indicates that these participants from the top 3 have at least one tag with which they would not want to work. The next step is also to take into account the workload of people according to the general estimation of tasks and the total number of tasks with the status to do.

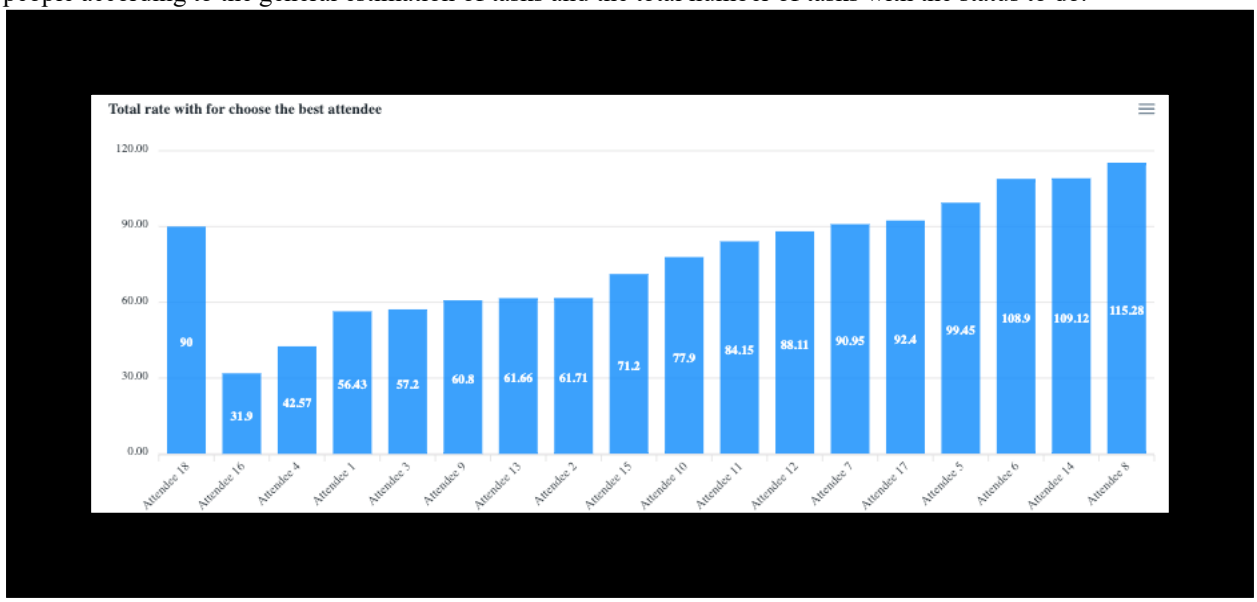

Fig. 10. Total rate for choose the best attendee

Taking into account the workload of people according to the general estimation of tasks and total tasks with status to do, we changed the rating and got one general rating as in Figure 8. From it we see that the candidates who were in the previous rating Attendee 5 - 117 points, Attendee 14 - 124 points, and Attendee 8 - 131 points again changed their position Attendee 5 to Attendee 6. Now the top 3 are occupied by participants Attendee 5 - 108.9 points, Attendee 14 - 109.12 points, and Attendee 8 - 115.28 points. As we see, the scores of all participants from the top 3 have decreased. This speaks of the fact that these participants from the top 3 have a certain number of tasks that affects the overall rating, but not significantly. And Attendee 5 is already loaded to a significant extent and cannot be given new tasks. Since there are no participants with the same scores, it can be said from Figure 8 that the most optimal option for task execution is Attendee 8. Since he has 115 points and he has a slight gap with participants Attendee 6 (108) and Attendee 14 (109).

Conclusions:

An automated information system has been proposed for determining the best candidate to perform a task, as well as for classifying the task and adding classification tags to it, which ensures accurate selection of the best performer of the task and performs accurate task classification and determines the corresponding tags. Further research is focused on increasing the accuracy and autonomy of the model and improving the level of classification to the level of subgroups of tags.

The prosed system of distribution and evaluation of tasks in the software development process is aimed at improving the accuracy and autonomy of the model, as well as improving the level of classification to the level of subgroups of tags. The system will use machine learning algorithms to identify the best candidate for a task, as well as to classify the task and add the corresponding tags. The system will also be able to identify the most suitable tasks for a particular candidate, based on their skills and experience. Additionally, the system will be able to provide feedback to the candidate on their performance, allowing them to improve their skills and become more efficient in their work. The system will also be able to track the progress of the task and provide timely updates to the stakeholders. Finally, the system will be able to provide detailed reports on the performance of the task and the candidate, allowing for better decision-making.

The proposed system of distribution and evaluation of tasks in the software development process for Scrum and Agile methods is designed to provide an automated and accurate way of assigning tasks to the best candidate, as well as classifying the task and adding the corresponding tags. The system is designed to be highly accurate and autonomous, and to provide a level of classification that is accurate down to the level of subgroups of tags. The system is also designed to be highly efficient and to provide a streamlined process for assigning tasks to the best candidate. The system is also designed to be highly flexible and to be able to adapt to changing requirements and tasks. Further research is focused on improving the accuracy and autonomy of the system, as well as improving the level of classification to the level of subgroups of tags. Additionally, research is being conducted to improve the system's ability to handle complex tasks and to provide a more efficient and streamlined process for assigning tasks to the best candidate.

References

1. Sudarkina S., Klimentova M., Anichkina I. Methods and Tools of Strategic Marketing Planning. / Sudarkina S., Klimentova M., Anichkina I. // Bulletin of the National Technical University "Kharkiv Polytechnic Institute" (Economic Sciences), (24), 2019. - Kharkiv: KhNTU, 2019. - Pp. 66-78.
2. Barkova K. Methodological Foundations of Strategic Planning of the Enterprise / Barkova K. // Recommended for printing at the meeting of the Scientific Council of the Kharkiv National Economic University named after S. Kuznetsov, Protocol No. 8 of May 3, 2019. - Kharkiv: KhNEU, 2019. - Pp. 41-49.
3. Kashtalyn I.V. Using Artificial Intelligence Systems to Improve Task Planning Tools / I.V. Kashtalyn, R.V. Kulinyak // III Scientific and Practical Conference of Young Scientists and Students "Intelligent Computer Systems and Networks". November 26, 2020, -Ternopil, Ukraine. Ternopil: ZUNU, 2020. - Pp.13-21.
4. Il'kov A.V. Features of Strategic Planning in Small Enterprises in Conditions of Uncertainty / Il'kov A.V. // Ensuring Sustainable Development of the Agricultural Sector of the Economy: Problems, Priorities, Prospects: Materials of the XI International Scientific and Practical Internet Conference October 29-30, 2020: In 2 vol. - Volume 1. - Dnipro: Publishing and Polygraphic Center "Garant SV", 2020. - Pp.72-74.
5. Ostapenko R.M. Organizational Mechanism of Strategic Planning of Agricultural Enterprises / Ostapenko R.M. // Materials of the II International Scientific and Practical Conference (online form) "Formation and Prospects of Development of Entrepreneurial Structures within the Integration into the European Space" - Poltava, 2019. - 715 p.
6. Kazimir V.V. Information Bases for Building Telecommunication Networks / V.V. Kazimir, V. Litvinov, S.M. Shkarlet, S.V. Zaitsev // Bulletin of Chernihiv State Technical University. - Chernihiv: ChDTU, 2013. - 340 p.
7. Steklov V.K. Information System: A Textbook for Students of Higher Educational Institutions in the Direction of "Telecommunications" / V.K. Steklov, L. Berkman. - K .: Technique 2014. - 792 p.
8. Stasev Yu.V. Computer Networks. Technologies and Protocols for Modeling: Tutorial / I.V. Ruban, S.V. Dudenko, O.I. Timochko. - Kh.: KhU P S, 2014. - 359 p.
9. Krivutsa V.G. Management of Telecommunications with the Application of Modern Technologies / V. Krivutsa, V.K.Steklov, L.N.Berckman, B.Kostik, B.Oliynyk, S.Sklyarenko // Textbook for Higher Education Institutions. - K.: Tekhnika, 2007. - 384 p.
10. Steklov V.K. Information System: Textbook for Students of Higher Educational Institutions in the Direction of "Telecommunications" / V.K. Steklov, L. Berckman. - K.: Tekhnika, 2014. - 792 p.
11. Krivutsa V.G. Management of Telecommunications with the Application of Modern Technologies / V.G. Krivutsa, V. Steklov, L. Berckman, B.Ya. Kostik, B. Oliynyk, S.M. Sklyarenko // Textbook for Higher Education Institutions. - K.: Tekhnika, 2007. - 384 p.
12. Gorbatiy, I.V. Telecommunications Systems and Networks. Principles of Operation, Technologies and Protocols: Tutorial / I.V. Gorbatiy, A.V. Bondarev // - Lviv: Publishing House of Lviv Polytechnic, 2016. - 336 p.
13. Sovetov, B.Ya. Modeling Systems: Textbook for Bachelors / B.Ya. Sovetov, S.A. Yakovlev. - 7th ed. - M.: Yurait Publishing, 2015. - 343 p.Chen S. Speaker, environment and channel change detection and clustering via the
14. Klimash M.M. Modern Transformations in Distributed System Architectures: Monograph / M.M. Klimash, A. Luntovsky, V. Romanchuk. - Lviv-Drohobych: Kolo, 2015. - 328 p.
15. Luntovsky A.O. Stages of Development of Modern Info-Telecommunication Services and Energy Efficiency of Network Technologies / A.O. Luntovsky, P. Gus'kov, A. Masiuk // Bulletin of the National University "Lviv Polytechnic". Series: Radioelectronics and Telecommunications. - Lviv: Publishing House of Lviv Polytechnic, 2014. - No. 796. - P. 131-139.
16. Brooks F. Mythical Man-Month, or How Software Systems Are Created. - St. Petersburg: Symbol-Plus, 2006. - 304 p.
17. Vendrov A.M. Design of Software for Economic Information Systems: Textbook. - 2nd ed., rev. and add. - Moscow: Finances and Statistics, 2005. - 544 p.: ill.
18. Iles P. What is Software Architecture? - Resource: http://www.interface.ru/home.asp?artId=2116
19. Kotzubay I.Yu., Chunaev A.V., Shikov A.N. Basics of Designing Information Systems: Tutorial. - St. Petersburg: ITMO University, 2015. - 206 p.

20. Minukhin S.V., Besedovsky O.M., Znakhur S.V. Methods and Models of Design Based on Modern CASE Tools. Tutorial. - Kharkiv: Vyd. KhNEU, 2008. - 272 p. (Ukrainian).
21. Pushchin M.N. Design of Information Systems: Tutorial. - Moscow: MIET Publishing House, 2008. - 234 p.
22. Design of Information Systems: Textbook / V.I. Grekul, G.N. Denischenko, N.L. Korovkina. - M. Internet-Un-t Information Technologies, 2005. - 304 p.
23. Reengineering of Business Processes. - Resource: https://library.if.ua/book/28/1899.html
24. Sommerville I. Software Engineering, 6th Edition.: Trans. from English. - M.: Williams, 2002. - 624 p.
25. Design Technologies. - Resource: https://studfile.net/preview/3997729/page:5
26. Fowler M. et al. Architecture of Corporate Software Systems. - M.: Williams, 2005. - 576 p.

Dmytro Okrushko **Дмитро Окрушко**	Master, Khmelnytskyi National University e-mail: okrushko@i.ua	магістр, Хмельницький національний університет
Antonina Kashtalyan **Антоніна Каштальян**	Ph.D, Associate Professor of Department of Computer Engineering and Information Systems, Khmelnytskyi National University e-mail: yantonina@ukr.net https://orcid.org/0000-0002-4925-9713	кандидат технічних наук, доцент кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет