INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 3, PP. 425-430
Manuscript received July 1, 2019; revised July, 2020. DOI: 10.24425/ijet.2020.131895

Improved Method of Searching the Associative Rules while Developing the Software

Tamara O. Savchuk, Natalia V. Pryimak, Nina V. Slyusarenko, Andrzej Smolarz, Saule Smailova, and Yedilkhan Amirgaliyev

***Abstract*—As the delivery of good quality software in time is a very important part of the software development process, it's a very important task to organize this process very accurately. For this, a new method of the searching associative rules were proposed. It is based on the classification of all tasks on three different groups, depending on their difficulty, and after this, searching associative rules among them, which will help to define the time necessary to perform a specific task by the specific developer.**

***Keywords*—software development, classification, C4.5 algorithms, associated rules, FPG-algorithm**

I. Introduction

THE one requirement among the important non-functional requirements for the modern comprehensive software is lifecycle requirement: identifying limitations on the human resources involved and the length of time during which such software is developed; and also the completion of the development process in time, which increases the market value of the final product. Therefore, an accurate estimation of the release date of the software product, based on the estimates of costs required for its development, is necessary for managers to plan the work of the team of software developers and more effective management of software projects [1,2,3,4].

One of the possible ways to verify that software is reliable is to tightly control all stages of its development. The traditional software development model requires testing of each development stage before moving on to the next one. Modern software projects are being developed more dynamically, by using more flexible technologies and methodologies; but the steps listed below are present in any process [5]:

1. Requirement analysis – at this stage, the area is being studied and the most important requirements for the future product are defined from the point of view of the customer or the user [6,7].
2. Prototype design – designing a user's requirement for a software product and specify requirements for the internal structure and operation of the future program from the point of view of programmers.
3. Development (coding) is the process of implementing a project using specific programming languages and specific tools.
4. Verification, testing and defect solving – the process of identifying and eliminating errors and establishing the compliance of the created product with its specification.
5. Documentation. During this process, documentation is created, the future product is described in terms of its creation, and in terms of its use [8,9,10].

The use of existing tools, such as program version control systems and database of tasks raised during the development of software, will allow collecting information about all changes in the code and use it for further analysis. Available databases contain a large amount of information that can not be manually analyzed, so there is a need for automated data acquisition and analysis. To solve this problem, such a method of data analysis is used as the search for associative rules, which allows you to find dependencies that can be used to predict the time required for the realization of a particular task by a particular developer [11,12,13].

II. Actuality of using data mining technologies, especially searching associative rules, in the software development process

The software is a computer program and data stored digitally and used to solve the defined tasks of a particular class [2]. Software development is a planned and defined process of creating programs [3].

One of the tools for managing software development is network planning, which is a management method based on the usage of the graph theory to display and algorithmize a set of interrelated works, activities or actions to achieve a specific goal [14].

The purpose of software development management is to minimize costs and adhere to development times. To achieve it, the following tasks must be solved [14]:

- Obtain the maximum material benefit that can be achieved by reducing to a minimum the duration of the software development process (by reducing the overall design and implementation time).
- Distribute the labor and resources involved as effectively as possible during the development of the software.

One of the main parameters of network planning is the critical path, which determines the timing of the entire planned work. The critical path is the longest chain of works leading from the initial event to the final event [14]. In the process of managing

Tamara O. Savchuk, and Natalia V. Pryimak are with Vinnytsia National Technical University, Vinnitsa, Ukraine, (e-mail: savchtam@gmail.com, nata.pryimak@gmail.com)

Nina V. Slyusarenko is with Kherson State University, Kherson, Ukraine, (e-mail: ninaslusarenko@gmail.com).

Andrzej Smolarz, is with Lublin University of Technology, Lublin, Poland, (e-mail: a.smolarz@pollub.pl).

Saule Smailova is with East Kazakhstan State Technical University named after D.Serikbayev, Ust-Kamenogorsk, Kazakhstan, (e-mail: saule_smailova@mail.ru).

Yedilkhan Amirgaliyev is with Institute of Information and Computational Technologies CS MES RK, Almaty, Kazakchstan, (e-mail: amir_ed@mail.ru).

software development, the focus is on the tasks that form the critical path. There are a lot of tools that help product managers with managing, designing, and assigning developers to the task, but they do not allow you to predict the duration of the task or project development based on the accumulated data. This lack of existing tools can be eliminated by developing the information technology of searching associative rules for the software development duration.

The large amount of information that appears during the software development process is stored by the companies in order to obtain useful data from it and a better understanding of this process and the final product. The information, stored in the repositories, can play a key role in improving the software development process and in the quality of the programs being developed [4,15].

Data Mining (DM) technology allows to analyze and obtain additional and interesting patterns from the information [4]. Different DM technologies can be used to solve the verity of tasks on different software process stages (figure 1), which are described above.

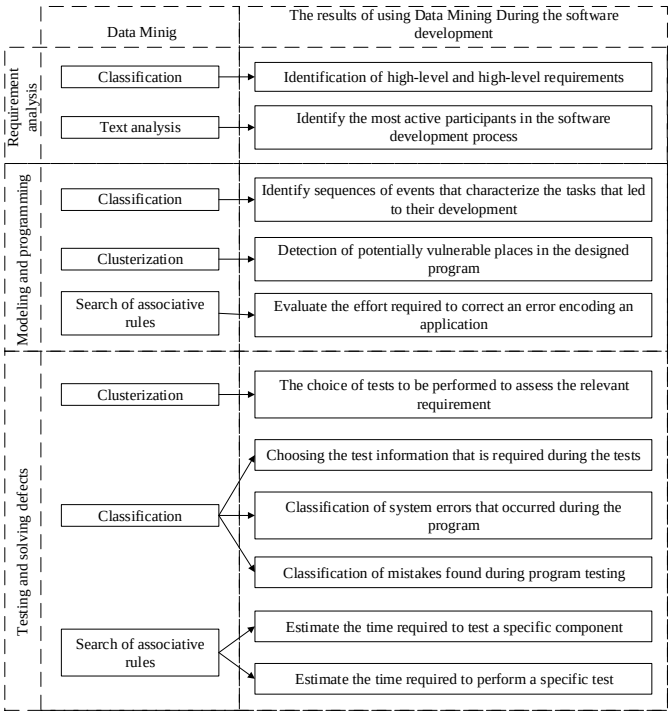

Fig. 1. The representation of using different DM technologies on different phases of software development

The main technologies of Data Mining, which can be used in software development, are [6, 9-14]:

1. The technology of intellectual analysis of textual information (correspondence, comments, documentation), which allows identifying the most active participants in the software development process.
2. A clustering technology that can be used to identify groups of similar modules of the program based on the number of modifications made; or to group the detected software defects according to specific components, which will help to more effectively correct these defects [15,16].
3. A classification technology that can help determine the mechanism for identifying vulnerable software modules based on the attributes of the module numbers.
4. Considerations based on precedents (case-based reasoning, CBR) will allow us to find similar cases in the past, analyze them and use this knowledge and information to a new problem. For example, when testing the software, namely after the program partition on the tested components, the CBR is used to find the required set of test instructions for a specific component, based on the data obtained when testing similar components before.
5. The technology of obtaining partial patterns or associative rules. An analysis of system errors that arose in the work of program modules can invent the relationship between these elements, based on the characteristic features of models and error categories.

Searching of associative rules as a very powerful technology can be used to solve or improve the next processes:

1. In the article [13] it is proposed to use associative rules to identify violations in the architecture of object-oriented software. This approach will help identify vulnerabilities in the software architecture in the early stages of its development, which will avoid additional material costs for their correction in the future.
2. The approach described in [14] will allow project managers to determine the amount of resources required and manage them effectively throughout the software development process by applying the theory of fuzzy sets and the search for associative rules using the Apriori algorithm, to re-evaluate the necessary human resources in the process of software development.
3. In research [15], authors are considering the use of associative rules search technology to determine the developer who will be assigned to fix the defect found in the program. Such an approach will allow automating the process of assigning a responsible person to correct a particular defect and can be used by project managers when planning the activities of the participants in the software development team.

Found during the software development the associative rules can be used by product managers, to organize the development process of high-quality software at a specified time and within the allocated budget, due to the use of found dependencies [18,19,20].

III. Aim of the Research

The purpose of the research is to search for more informative associative rules, during the development of software, and at the same time to increase the speed of this search.

Objectives of research:

1. To develop the information model of the process of searching the associative rules.
2. Improve the method of searching associative rules FPG, by using classification to divide tasks depending on their difficulty.

IV. Solution

In a result the associative rules, which allow to find relationships between related events or elements [17] and can be described like $X \to Y$, $X \cap Y \to \varnothing$ will be found.

Any association rule can be described by two main characteristics [18,19]:

1. Support $supp(X \to Y)$ of the associative rule $X \to Y$ is a value equal to the ratio of the number of records $X \cup Y$ in

the database D, to the total number of records in the database.

2. The confidence of $conf(X \to Y)$an associative rule $X \to Y$ is a value that is equal to the ratio of its support $supp(X \to Y)$ to the support $supp(X)$ of the set X.

When searching the most informative associative rules it is necessary to find the set of all associative rules AR in which the value of support is higher than minimum support $supp_{min}$ set by an expert manually. It is reflected in the expression:

$$AR = \{D | supp(X \to Y) > supp_{min}\}$$

To define the only interesting and valuable associative rule for the expert confidence $conf(X \to Y)$ of these associative rues is used. The value of the minimum confidence $conf_{\min}$ is given by an expert and numerically greater than the value of the minimum support:

$$supp_{\min} < conf_{\min}$$

The set of useful associative rules UAR is a subset of the set of all possible associative rules AR and can be described by:

$$UAR = \{AR \mid supp_{\min} < conf_{\min}\}$$

The information model of the search process of the associative rules when developing the software displays its input and output values, as well as their relationships, and can be described as a tuple:

$$IMARM = \left\langle \begin{matrix} Task, Dev, minsupp, minconf, \\ method, UAR, newTask, PredictTask \end{matrix} \right\rangle$$

where *Task* – set of tasks, that were done during software development; *Dev* – set of developers, which can perform the tasks; *minsupp* – minimal value of associative rules support; *minconf* – minimal value of associative rules confidence; *method* – method, that will be used to search associative rules; *UAR* – set of found associative rules; *newTask* – set of tasks, for which it's necessary to predict the time, necessary to perform them by developer; *PredictTask* – set of tasks, for which the time, necessary to perform them by specific developer, is defined.

The determine scheme of the information model of the process of finding and using associative rules in software development is shown in figure 2.

Each $Task_i$ is described by:

$$Task_i = \langle Type, Priority, Severity, Component, t, Dl, Cx, Sx, Rl \rangle,$$

where $i = \overline{1,n}$, n – number of tasks; *Type* – type of the *i*-th task; *Priority* – priority of the *i*-th task; *Severity* – severity of the *i*-th task; *Component* – component of the developed software; *Dl* – level of the developer skills, who will perform this *i*-th task; *t* – time, necessary to perform the *i*-th task by specific developer, *Cx* – complexity of the *i*-th task, that needs to be resolved; *Sx* – the gender of the developer who completed the task; *Rl* – release number during which the *i*-th task was performed. Each of these elements has a set of values, which vary depending on the project.

As not all characteristics of a task equally affect the duration of its development, multivariate correlation analysis was performed to determine the degree of correlation between the characteristics of the tasks performed during software development and the duration of their development.

As the set of possible values for the characteristics of the tasks (software component, developer qualification level, task type, complexity, priority, severity, developer gender, and software release) are qualitative, Chuprov Mutual Coefficient was calculated to determine the set of tasks characteristics, which will be used after during the analysis.

After the performed calculation the closest was got between the duration of development and the following characteristics: the level of the developer skills; the type, severity, complexity, priority of the task, and the component of the software. The relationship between the duration of the task development and the release during which the task is performed is lower than other characteristics, so this factor will not be taken into account when searching for the associative rules for the software development duration

Also, was noticed that the closest link exists between the developer's gender and the skill level, which means that these two characteristics are redundant. If compare the impact of each characteristic on the duration of the task, it can be seen that the skill level has a higher correlation level than the developer's gender, and therefore should be used when looking for associative rules for the software development duration.

Therefore, the set of the task characteristics, which are taken into account while searching the associative rules for the software development duration, are as follows: the level of developer skills; type, criticality, complexity, and priority of the task; component of the software being developed.

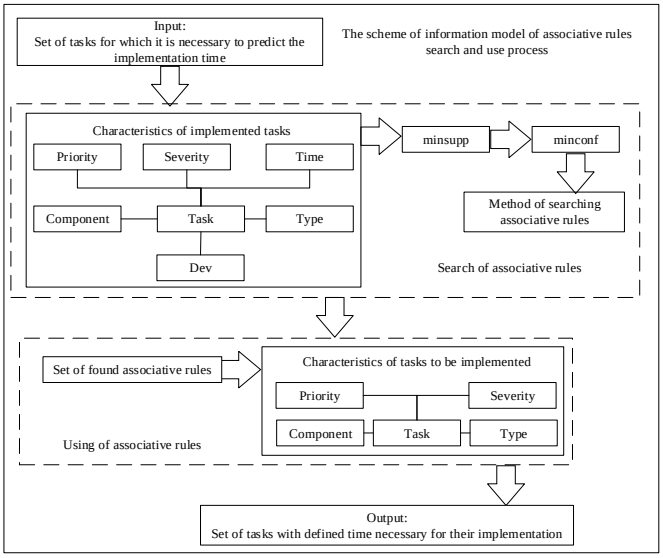

Fig. 2. The scheme of information model of associative rules search and use process

The set of found associative rules is described next:

$$AR_d = \left\{Task_i \cap Dev_j \to t_{i_j}, Task_i \cap Dev_j \cap t_{i_j} \to \emptyset\right\},$$

where $d = \overline{1,c}$ d is the number of associative rules, Dev_j is the name of the *j*-th developer, who will perform this *i*-th task, t_{i_j} is the time, necessary to perform the *i*-th task by *j*-th developer.

Method Frequent Pattern Growth (FP-Growth, FPG) was selected for generating frequent patterns to search the associative rules in software development among them. The performed analysis of comparing and choosing the most appropriate method is described in [20]. The brief analysis is also represented in Table I.

TABLE I
ANALYSIS OF THE METHODS OF SEARCHING ASSOCIATIVE RULES

Method	Advantages	Disadvantages
Set-oriented	Processing large databases, easy to understand.	The need for a large amount of time, space and memory of the computer for the process of generating possible frequent candidates, multiple-time database scanning, the ability to work only with static data.
Partition	The ability to quickly search in large databases, only two database scans, reducing the use of the disk i / o.	The necessity to have a computer with a large amount of operating memory, an imperfect approach to dividing the database into parts in the first stage
Itemset Clustering	Using not a lot of RAM, the lack of building a hash structure, only one database scan.	The need to select parameters for the clustering algorithm.
FP - Growth	The size of the FP tree is a small, high-speed search for frequent object sets.	The size of the FP tree can not be larger than the main memory of the computer, the existing modifications of the method do not take into account the peculiarities of the software development process, which may negatively affect the finite set of associative rules found.

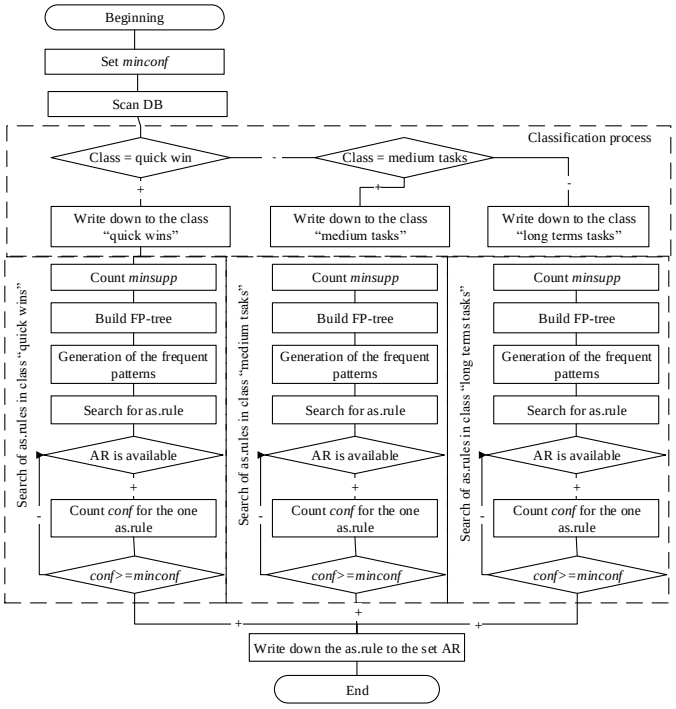

Fig. 3. The scheme of information model of associative rules search and use process

Using the FP-Growth method of searching associative rules in software development allows you to find dependencies in the database without generating candidates for frequent patterns, but has the following disadvantages [21,22]:

- the size of the FP-tree, can not exceed the size of the main memory of the computer;
- the existing modifications of the method do not take into account the peculiarities of the software development process, which may negatively affect the finite set of associative rules found.

The identified disadvantages can be eliminated by improving the FPG method (figure 2), which will improve the quality of the found associative rules. Modification of the method consists in the fact that there will be a classification of the entire set of tasks into three groups, depending on the parameters of the task:

- quick wins (tasks which are easy for their realization);
- difficult tasks (tasks which are difficult for their realization);
- medium tasks (tasks which are between quick wins and difficult tasks).

Each group will be analyzed separately, which will increase the efficiency of the identified APs. In turn, the parallel processing of these three groups will reduce the time for defining frequent subject sets, which will speed up the implementation of the method [23,24].

For the classification of the tasks method of decision trees was selected and algorithm C4.5 will be used. As the study sample is not very large and it's necessary to get easily interpreted data in a short time, then this method is the most appropriate to use. The disadvantage of this algorithm, which is slightest fluctuations in the data can lead to the formation of different decision trees, won't have an influence on the results of the classification [25,26,27].

Found associative rules of software development duration can be used to predict the time required for task completion by a developer with a certain level of skills. The resulting information can later be used by project managers to manage the software development process and to build a plan for the effective use of human resources.

Formally, the algorithm for using associative rules for the duration of software development consists of the following basic steps:

1. There is a load of tasks for which it is necessary to determine the duration of development by the developer.
2. The first task of the set is selected.
3. An associative rule is selected from the set of associative dependencies with which the characteristics of the task will be compared.
4. There is a comparison of the values of the characteristics of the tasks from the set of associative rules and the new task.
5. If the values of the characteristics of the tasks do not match, then the next associative rules are selected from the found set, and the values of the characteristics described in step 2 are compared.
6. If the characteristics are the same, then the duration of task development by the developer with the specific level of skills is determined for this task. This information is derived from the relevant associative rule.
7. Obtained data time and skill level of a developer is added to a task that is saved into a plurality of time-bound tasks. The information from it is subsequently used to determine the duration of software development and build an optimal software development plan.
8. If the characteristics of all ARs do not coincide with the values of the characteristics of the new task, it is recorded in a set of tasks for which the duration of the development is not determined.

9. For tasks in this set, the duration of development is determined by a developer survey. After determining the time required for their development by a developer of a certain qualification, they are also added to the set.

As the software development process is dynamic and most of the modern software is developed using Agile technologies, namely Scrum. One of the key elements in Scrum is the amount of time the team creates the part of the program that can be presented to the customer – "sprint". During such intervals, there is an accumulation of newly developed tasks in the database, so there is a need to search for new possible ARs after each sprint. This approach will allow more accurately determine the duration of the task by the developer of a certain qualification since new ARs on the duration of software development will be formed on the basis of a more powerful sample of data.

V. Results of the Experiments

The experiment was performed to find the results of the modified method of searching associative rules during the software development process. The big software project PCOUNTER was selected for research. All tasks for this software are stored in the managing tracking systems.

The number of found associative rules and time spent on their search are represented in table 2. The experiment was performed 3 times for the different total number of tests in the database (DB).

TABLE II
NUMBER OF FOUND ASSOCIATIVE RULES AND TIME SPENT ON THEIR SEARCH

Method	Experiment 1		Experiment 2		Experiment 3	
	original FPG	modified FPG	original FPG	modified FPG	original FPG	modified FPG
Number of tasks	100		500		1000	
Time spent to find associative rules (sec)	1.988	1.981	6.258	6.184	12.754	12.502
Number of associative rules found	10	9	32	27	57	48

The value of minimum support was different depending on the total number of the tasks in the database and for minimum confidence, the value was 1 higher than minimum support. The results of this study are also shown in the plot in figure 3.

As it is possible to see from the plot the time necessary for the search of the associative rules in increasing depending on the total number of tasks in the database. Also, the difference between the time spent for the search of associative rules during the software development by using the original FPG method and modified is bigger with the increasing of the total number of the test. As a result: it's more useful to use the modified algorithm when the total number of tests is big (more than 800 items in DB).

The difference between found associative rules with the help of the original FPG method and modified method is also visible, which means that associative rules are more efficient and valuable for the expert.

A study was also conducted on the relationship between the number of completed tasks during software development and the number of ARs found among them. Also, research about the accuracy of determining the duration of the task using found rules was performed.

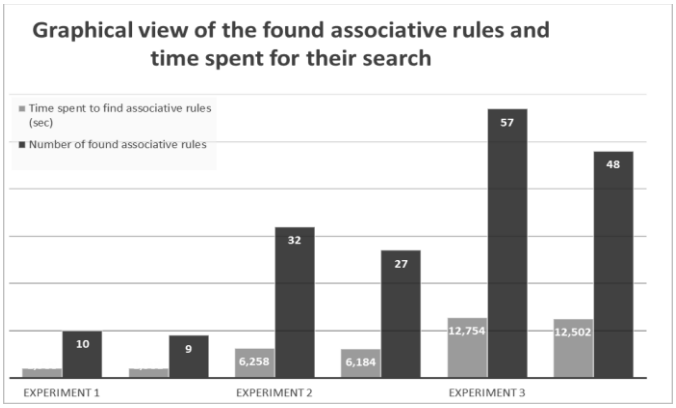

Fig. 4. Graphical view of the found associative rules and time spent on their search

As can be seen from figure 4 with the increasing number of completed tasks stored in the database, the number of associative rules found among them is also increased. Thus, among 100 tasks, 9 ARs were found, and among 1000 tasks – 48 rules, which means that when the number of tasks increased by 20 times, the number of ARs found increased more than 5 times, respectively.

In turn, with the increasing number of associative rules, the accuracy of determining the duration of the task, characterized by error, increases. The simulation results are shown in figure 5.

With 14 associative rules found, the accuracy of determining the duration of the task was 12%, while increasing the number of rules to 153, the accuracy increased 2.3 times (5.2%), which allowed managers to plan the software development process more efficient.

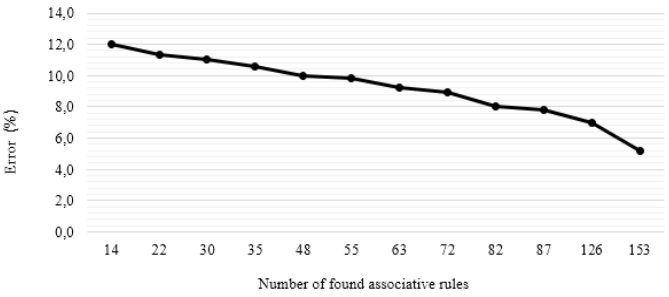

Fig. 5. Dependency between the accuracy of determining the tasks implementation time and the number of associative rules found during the software development process

VI. Conclusions

1. There was done an analytical review of the possibility to use data mining technologies during software development, and it also was defined that the technology of searching associative rules is very powerful and can help to solve different tasks during the software development process. The information model of finding associative rules during software development was developed.
2. Method of searching associative rules FPG was modified, by using classification algorithm C4.5, which will classify the tasks to the three groups depending on their difficulty. Due to what, the time of searching associative rules is less compare to the original algorithm and the efficiency of associative rules is improved.

3. To verify the functioning of the modified method FPG of finding associative rules during software development, the experimental studies were performed. They showed that the speed of searching associative rules using the modified algorithm is increasing with the increment of the total number of tasks in DB. The number of found associative rules is less compare to the original method FPG, but it means that they are more efficient and valuable for the expert, which can use them to define the time necessary to perform the specific task by specific dev.
4. Also, the result of the research showed the accuracy of determining the time, needed to implement the task, increases by 2.3 times, with increasing the number of found associative rules by 11 times, which proves the expediency of finding such dependencies for the software development duration.

REFERENCES

[1] K. Wiegers, *Software Requirements 3rd Edition*, NY, USA: Microsoft Press, 2013.

[2] V. K. Batovrin, *Explanatory dictionary on system and software engineering*, Moscow: Press, pp. 280, 2012.

[3] S. Wang, D. Samadhiya, and D. Chen, "Software Development and Quality Problems and Solutions by TRIZ," in *International Symposium on Frontiers in Ambient and Mobile Systems*, Ontario, 2011, pp. 730 – 735.

[4] M. Halkidi, D. Spinellis, G. Tsatsaronis, and M. Vazirgiannis "Data mining in software engineering," in *Intelligent Data Analysis*, Amsterdam, 2011, pp. 413 – 441.

[5] S. J. Greenspan, and C. L. McGowan, "Structuring software development for reliability", in *Microelectronics Reliability*, vol. 17, no. 1, pp. 1987, pp. 75 – 83.

[6] T. Xie, S. Thummalapenta, D. Lo, and C. Liu, "Data Mining for Software Engineering," in *Institutional Knowledge at Singapore Management University*, Singapore, 2009, pp. 55 – 62.

[7] A. A. Barseagian, M. S. Kuprianov, V.V. Stepanenko, and I. I. Holod., *Data mining technologies: Data Mining, Visual Mining, text mining, OLAP*, Saint-Petersburg, 2007, pp. 384.

[8] I. A. Chubukova, "Data Mining," in *Internet-University of Information Technologies, Binom. Knowledge lab*., 2008, pp. 384.

[9] D. Hand, H. Mannila, and P. Smyth, "Principles of data mining (adaptive computation and machine learning)," in *Cambridge: The MIT Press*, 2001, p. 546.

[10] A. Berson, and S. Smith., "Data Warehousing, Data Mining and OLAP," 2007, pp. 612.

[11] T. O. Savchuk, and N. Pryimak., "Using the methods of data mining intelligence when testing software," *Intelligent Information Technologies*, pp. 87, 2016.

[12] M. Sharma, and M. Kumari, "Bug Assignee Prediction Using Association Rule Mining," *Springer*, pp. 444 – 457, 2015.

[13] C. Maffort, M. Valente, and M. Bigonha, "Mining Architectural Patterns Using Association Rules," in *International Conference on Software Engineering and Knowledge Engineering (SEKE'13)*, Boston, 2013.

[14] D. White, and J. Fortune, "Current practice in project management – an empirical study", *International Journal of Project Management* , vol. 20, no. 1, pp. 1 – 11, 2002.

[15] M. Azzeh, D. Neagu, and P. Cowling, "Software Stage-Effort Estimation Using Association Rule Mining and Fuzzy Set Theory," Badford, pp. 154 – 160, 2010.

[16] M. Sharma, M. Kumari, and V. Singh, "Bug Assignee Prediction Using Association Rule Mining," in *Department of Computer Science University of Delhi*, Delhi, 2015, p. 445.

[17] T.A. Zayko, A. A. Oliinyk, and S. A. Subbotin, "Association rules in data mining," *Herald of the National University "KhPI"*, no. 39 (1012), pp. 82 – 96, 2013.

[18] C. Zhang, and S. Zhang, "Association Rule Mining, Models and Algorithms," pp. 244, 2002.

[19] T.O Savchuk, and N. Pryimak., "Modeling of software development process with the markov processes," *Eastern-European journal of Enterprise technologies*, pp. 33 – 38, 2017.

[20] T.O Savchuk, and N. Pryimak., "Submission of the selection of the method of generation of particular subjects for search of associative rules for the project development of the programmable supply," *Internet, Education, Science*, pp. 43 – 44, 2018.

[21] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *SIGMOD*, vol. 29, pp. 3 – 12, 2000.

[22] M. Kacprowicz, "An interval type-2 fuzzy systems in the management of emissions of nitrogen oxides," *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska – IAPGOS*, vol. 5, no. 1, pp. 20–23, 2015.

[23] V. Vassilenko, S. Valtchev, J. P. Teixeira, and S. Pavlov, "Energy harvesting: an interesting topic for education programs in engineering specialities," *Internet, Education, Science*, pp. 149-156, 2016.

[24] M. Górecka, K. Górecki, "Comparison of selected tools for computer analysis of digital circuits," *Przeglad Elektrotechniczny*, vol. 94, no. 4, pp. 72–75, 2015.

[25] A.P. Rotshtein, and H.B. Rakytyanska, "Diagnosis problem solving using fuzzy relations," *IEEE Transactions on Fuzzy Systems*, vol. 16, pp. 664-675, 2008.

[26] L. I. Timchenko, "A multistage parallel-hierarchic network as a model of a neuronlike computation scheme," *Cybernetics and Systems Analysis*, vol. 36, pp. 251-267, 2000.

[27] L. I. Timchenko, Y. F. Kutaev, V. P. Kozhemyako et al., "Method for training of a parallel-hierarchical network, based on population coding for processing of extended laser paths images," *Proceedings of SPIE 4790*, 2002.