Designing Risk Resilient Networked High-Load Computing Web-Systems for Information Flow Processing

Nadiia Pasieka[1], Zora Říhová[2], Marta Vohnoutová[2], Nelly Lysenko[1], Oleksandra Lysenko[1], Vasyl Sheketa[3], Mykola Pasyeka[3] and Nataliia Kulchytska[1]

[1] *Vasyl Stefanyk Precarpathian National University, Ivano-Frankivsk, 76000, Ukraine*
[2] *University of South Bohemia, Branišovská 1645/31a, České Budějovice, 370 05, Czechia*
[3] *National Tech. University of Oil & Gas, Ivano-Frankivsk, 76068, Ukraine*

Abstract

Improved models and algorithms for the architecture of a high-loaded risk resilient Web-system, whose main differences are the possibility of aggregation and sharing of large sets of heterogeneous computing resources to process information data distributed between geographically separated territories. The proposed model and algorithms allow the efficient and secure use of additional network resources connected to the functional network, in contrast to traditional approaches, when these resources are not available within a single computing node on an independent computing platform. Subsequently, innovative approaches have been developed to build high-load risk resilient distributed cluster software systems, which provides a significant increase in the total amount of effective processing of information flows of the node communication system as a whole. Therefore, the use of this approach is appropriate for distributed risk-tolerant software systems where rapid loss of information flows is highly undesirable. Next, algorithms were developed for automated load management of independent computer platforms of information data flows for efficient scaling (clustering) of risk-resistant software systems.

Keywords

Risky software systems, task distribution, algorithms, architecture, Web-systems.

1. Introduction

Balancing the load on computing nodes provides an even load of hardware and software systems on independent computing platforms. A risk-resistant software system that balances the computational load must automatically decide on which node to perform the affective calculations associated with the new information task. So, the main task of balancing is the process of effective support of the process of transferring (migration) part or the whole calculation from the most loaded computing

CITRisk'2021: 2nd International Workshop on Computational & Information Technologies for Risk-Informed Systems, September 16–17, 2021, Kherson, Ukraine
EMAIL: pasyekanm@gmail.com (N.Pasieka); zora.rihova@trilogic.cz (Z.Říhová); marta.vohnoutova@gmail.com (M.Vohnoutová); nelli.lysenko@gmail.com (N.Lysenko); lysenkowa@gmail.com (O.Lysenko); vasylsheketa@gmail.com (V.Sheketa); pms.mykola@gmail.com (M.Pasieka); nataliia.kulchytska@pnu.edu.ua (N.Kulchytska)
ORCID: 000-0002-4824-2370 (N.Pasieka); 0000-0003-3896-4297 (Z.Říhová); 0000-0002-8915-8626 (M.Vohnoutová); 0000−0002−1029−7843 (N.Lysenko); 0000−0002−1029−7843 (O.Lysenko); 0000−0002−1318−4895 (V.Sheketa); 0000−0002−3058−6650 (M.Pasieka); 0000-0001-9308-6840 (N.Kulchytska)

nodes to the less loaded ones, input factor loading i of $i = 1, ..., r$; u_j – output parameter weighting, output parameter weighting j of $j = 1, ..., s$.

When solving the problem of risky maximization of an efficiency criterion, the actual problem of having a share in the distribution of two linear aggregate quantities arises [46].

Also, the problem of maximizing the efficiency of risk-sustainable software Web-systems is called linear particle programming. At the same time, there are many possibilities for transforming linear particle programming into a linear programming problem.

$$f_0 = \frac{\sum_{i=1}^{r} v_i x_i}{\sum_{j=1}^{s} u_j y_j} \to min!, \quad (1)$$

with $\frac{\sum_{i=1}^{r} v_i x_i}{\sum_{j=1}^{s} u_j y_j} \geq 1$– or all modules $m = 1, 2, ..., n$; $u_j \geq 0$, $j = 1, 2, ..., s$; $v_i \geq 0$, $i = 1, 2, ..., r$; y_j – expression j – and the output parameter of the investigated computational module (node); x_{im} – expression i – that input factor m – of that computing node with $i = 1, ..., r$ and $m = 1, ..., n$; y_{jm} – expression j – that input factor m – of that computing node with $i = 1, ..., r$ and $m = 1, ..., n$; v_i – input factor weights i with $i = 1, ..., r$; u_j – weighing the output parameter j with $j = 1, ..., s$.

For further investigation we modify this nonlinear problem of risky criterion optimization using a complex fractional programming theory algorithm, a more traditional and less resource-intensive linear programming algorithm. Thus, the complex problem of optimizing the risk-sustainable efficiency criterion can be simplified to a linear problem using linear optimization techniques [28]. To obtain the efficiency criterion value for all computational modules (nodes), it is necessary to solve the maximization problem individually for each individual module involved in scientific research. In this case, the vectors xim and yim are each time replaced by the profile of input and output parameters of the computational module under study, respectively [24]. In the other problem of maximizing the risky criterion, they remain the same for each computational node [18, 30, 35, 45]. In this problem, constraints are imposed to ensure that qualitative efficiency values are found in the region between zero and one ($e_0 \in [0,1]$).

The main property of this model is formed from the constraint criterion of risky sustainable efficiency. Its target function proportionally tends to increase the output parameter of the observed computational module to the limit of the risk-sustainable efficiency criterion. When a certain function is mathematically decoupled, after applying the duality theorem to it, an equivalent or so called encompassing form is formed. The principle of duality when using linear programming indicates that for each considered direct linear model of optimization of the risk-sustainable efficiency criterion there exists a dual corresponding linear optimization model, and in the case of solving one model contains all expressions for solving the other model. Thus, the original optimized model will be treated as a direct linear programming problem. Using the dual method, we minimize the weighted sum of the input parameters in one normalized output parameter. Mathematical formulation of linear programming model when using dual method is carried out as follows:

$$\min g_0 = \sum_{i=1}^{r} t_i x_i, \quad (2)$$

with g_0 – the value of the efficiency criterion of the studied computing node; x_i – expression i – of that computing node input factor with $i = 1, ..., r$; t_i – variable weighting factors.

Linear combinations determine potential reference groups for measuring the efficiency criterion of computational nodes involved in the study. So, according to the task of optimization (2) of the target function which is minimized, the proposed method selects such a reference group in which the criterion of efficiency of the involved computational node looks not effective enough [10, 37, 40].

Therefore, this mathematical problem can be interpreted as follows: for the investigated computational node to determine the minimum efficiency criterion of the input parameter G_0, where compared to the weighted probabilities of comparative units, the weighted combination of output parameters of any input

parameter does not detract from the output parameter, and the weighted integral combination of input factors of any input factor is G_0 times greater than the input factor.

2. Parametric network model with time metrics to calculate load volume

The basic principles of operation and functioning of the network model of cloud computing on independent computing platforms differ significantly from traditional serial and parallel models. The main feature of the network model of cloud computing is the ability to aggregate and collectively use large information sets of heterogeneous computational data flows, distributed geographically. Basically, this approach provides significant advantages, for example, when a software system is developed that requires information resources that are not available within one computing node, it can get them in other computing nodes connected to the cloud network [5, 8, 9, 23, 27].

However, the use of such a complex architecture for processing data flows has several caveats and some problems. To a highly heterogeneous, dynamically formed distributed computing environment it is rather difficult to apply such traditional metrics of performance criterion definition as the speed of data flow calculation, bandwidth of exchange channels, etc. to such a highly heterogeneous, dynamically formed distributed computing environment. That's why to estimate the quality of the provided cloud service it's necessary to use specialized computational metrics [11, 13, 31]. Let's assume that in a network cloud environment m computing resources are available and there is a system of distribution of task flow t, which provides an even distribution of information tasks $j \in t$ into available resources. Within the framework of using such developed software system with the use of cloud technology each user's information task can be divided into certain computing actions $k \in j$. When setting a task, the processing time d_j is determined in order to get the corresponding junction. Each information task of user j and all corresponding actions $k \in j$ are in the cloud grid and at a certain point r_j. So, a cloud network that operates in online mode, with custom r_j values that are unknown in advance for a significant number of these tasks. As soon as a certain custom task for processing arrives in the cloud network, it is planned to search for and allocate the necessary resources to run it. Suppose that as a result of the final distribution of computing tasks S for each action $k \in j$ a certain time for processing $C_k(S)$ is required. So, a user's computational task in the cloud network j can be processed not faster than in a certain period of time determined by the expression (3):

$$C_j(S) = \max_{k \in j} C_k(S), \tag{3}$$

with $C_k(S)$. – the maximum processing time of the user's task; k – action; j – mission.

Let us define p_j as the processing time of user's task $k \in j$. So, the processing time of user's task in a cloud network can be calculated as follows (4):

$$p_j = C_j(S) - \min_{k \in j}(C_k(S) - p_k), \tag{4}$$

with p_j – user task processing time; $C_k(S)$ – execution time; $C_j(S)$ – maximum execution time; p_k – decision implementation time.

The received integral values allow estimating properties of the cloud network computing environment. To analyze the quality of the cloud service provided by the network computing environment, you can use the value of the maximum delay in user tasks (5):

$$L_{max} = \max_{j \in t}\big(C_j(S) - d_j\big), \tag{5}$$

with L_{max} – maximum delay of the user's task; $C_j(S)$ – maximum processing time; d_j – time of uncompleted user tasks.

In order to optimize the work of a distributed cloud, it is necessary to achieve the minimization of L_{max} value, and you can also use the T_j value, which determines how late the user's task is (6):

$$j \in t \wedge C_j > d_j, \tag{6}$$

with t – information computing resources are available; d_j – time of uncompleted user tasks; C_j – time to complete one task; k – action; j – mission.

Consequently, this indicator provides information on the number of outstanding computational requests from users who have entered the cloud network for processing. Consumption of computing resources of the RC_k cloud network is a certain subtask of computing, which is defined as the product of the corresponding solution time by the number of resources used in the network (7):

$$RC_k = p_k \times m_k, \tag{7}$$

with RC_k – total consumption of network computing resources; p_k – decision implementation time;
m_k – the number of resources used.

Using the value of integrated consumption of computing resources of the cloud network, you can calculate the value of available information resources U (8):

$$U = \frac{RC(S)}{m \times (\max\limits_{j \in t} C_j(s) - \min\limits_{j \in t}(C_j(S) - p_j)}, \tag{8}$$

with U – the amount of use of available information resources; m – resource quantity; p_j – processing time of the j-th problem; $RC(S)$ – time of total consumption; $C_j(S)$ – time of the j-th task.

By a certain value that characterizes the criterion of optimally used computing resources of the cloud distributed network. In the process of processing information requests in the cloud distributed network, there are often situations when the software system fails during the processing of the user's task [6, 7, 26, 43, 44, 47-48].

Then the user's task to process the information request must be run several times for its successful execution. Because users and administrators may have different (and even conflicting) requirements for a cloud-based Web system, it is difficult to find the right metrics that are universal and satisfying for everyone. From the user's point of view of the developed software system using cloud technology, the following metrics of the average response time to the request (Average Response Time) and the average waiting time of the request (Average Wait Time) can be distinguished:

$$ART = \frac{1}{|t|}\sum_{j \in t}(C_j(S)), \tag{9}$$

with ART – average response time; $C_j(S)$ – time of the j-th task; p_j – time of realization of the j–th task; t – available resources.

$$AWT = \frac{1}{|t|}\sum_{j \in t}\left(C_j(S) - p_j\right), \tag{10}$$

with AWT – average waiting time; $C_j(S)$ – time of execution of the j – th task; t – available resources.

The ART parameter value characterizes the average response time to a user's information request, namely, how quickly user tasks are processed. Besides, value of parameter AWT is very important for developers of algorithm of actions on rather small information tasks, and also on processing of their inquiries. So, the optimum and simple method of measurement of fair efficiency of use of information and hardware resources is calculation of deviation of weighted average time of expectation of processing of inquiry:

$$AWTD = \frac{1}{|t|}\sqrt{\sum_{j\in t}(C_j(S) - p_j)^2 - (\sum_{j\in t}\frac{C_j(S) - p_j}{|t|})^2}, \quad (11)$$

with *AWTD* – deviation of the average waiting time for processing the user's information request; *t* - available resources; $C_j(S)$ – time of execution of the *j* – th task; p_j – time of realization of the *j* – th task.

AWTD must be minimized in order to achieve optimal results in the cloud network. In modern cloud network Web-systems, the ability to complete processing of a given volume of tasks is more important than the acceleration of distributed visco-productive Web-system, obtained with this approach to processing. It should be noted that the information tasks of the user, which are carried out in cloud network environments, can have a rather complex architecture than the information tasks of the user, which are carried out in parallel systems with traditional architecture. For example, the information flows of user tasks have a more complex logical structure than the packages of corresponding tasks [25]. Using a cloud mesh requires changing such notions as errors of a developed program system of Web-services which is designed on a mesh model, generates error messages as soon as there comes a situation when it is impossible to successfully execute and finish a user's information task. For example, a failure of a developed software system using cloud technology may occur if it is impossible to find the appropriate resources to perform information calculations or through their completion.

Using the concept of fault tolerance in software systems using cloud technology, we can define as the main possibility to increase the time delay of both software and hardware errors until there is no chance that the work on processing user requests will be successfully completed. Metric of completed user request processing in a software system using cloud technology Workload Completion, which is formed as a ratio of successfully completed user tasks to the total volume of all requests set by the cloud network scheduler:

$$WC = \frac{\sum j \in t \wedge (j\ complited)}{|t|}, \quad (12)$$

with *WC* - indicator of completed user requests processing; *t* – available resources; *j* – task.

This metric allows to define the main limitations of the cloud network software system, and its maximization can be the main goal. However, it also has some critical limitations from the point of view of using free information and hardware resources, as a user's task with a smaller number of computational actions has a significant influence on the changes of this value [16].

Task completion calculates the number of completed actions to the total number of performed actions, which were implemented within the cloud software system of user tasks distribution:

This metric allows you to define the main limitations of a cloud network software system, and its maximization can be the main goal. However, it also has some critical limitations from the point of view of using free information and hardware resources because the task of a user with less computational actions has a much greater influence on the change of this value.

Task completion calculates the number of completed actions to the total number of performed actions, which were implemented within the cloud software system of user tasks distribution:

$$TC = \frac{\sum j \in t \wedge k \in j \wedge (k\ complited)}{\sum j \in t\ |j|}, \quad (13)$$

with *TC* – index of completed actions for processing user tasks; *t* - available resources; *k* - action; j - task.

It is also worth considering such a notion as the completion of unlocked actions from user tasks for processing enabled task completion, i.e. when they can be performed only when all corresponding dependencies for a given sequence of actions will be executed by a software system using cloud technology:

$$ETC = \frac{\sum j \in t \wedge k \in j \wedge (k\ complited)}{\sum j \in t \wedge k \in j \wedge (k\ enabled)}, \quad (14)$$

with *ETC* – integral indicator of completed unlocked actions relative to user tasks; *t* – available resources; *k* – action; *j* – task.

So, we have considered the main principles of the cloud network model's functioning while developing program systems which differ greatly from traditional serial and parallel models. Their main difference is the possibility of aggregating and sharing large sets of heterogeneous information resources distributed between geographically separated computational independent platforms. Basically such architectural approaches to developing program systems with the use of cloud technology bring considerable advantages and financial benefits, for example, when a program Web-system requires considerable information resources which are inaccessible within one computing node, it can get them in another node connected to the cloud network.

3. Models and algorithms of distributed high-performance software Web-system architecture optimization

Any Web-systems or cloud services that serve a lot of users, a priori highly loaded. However, highly loaded distributed software Web-systems cannot effectively apply models, methods and algorithms that are used as basic approaches to the development of ordinary Web-sites [4, 42]. A considerable increase of the user audience and corresponding calculations without proper approaches to optimization of the architecture of program systems development with the use of cloud technology can lead to considerable complications in their maintenance in time [1, 17, 21, 29]. Now all the models, methods and algorithms for developing the architecture of distributed fault-tolerant Web-systems are developed and used with the use of generally accepted information technologies [19].

Methods of sending information packets should use an algorithm for calculating the checksum of the control protocol transfer of the package, which allows you to display the checksum of the control protocol transfer of data of the package using the checksum of the header of this package [2, 12, 20, 22, 36].

Having considered the checksum recalculation algorithm, which is to use the output package checksum and the input package checksum to calculate the checksum. So, if the clipboard is divided into two information parts, the checksum of the entire buffer can be linearly expressed through checksums of its parts. Typical length of the header protocol control transmission of the package, usually has a shorter length by several times than the basis of the entire information package. Consequently, the computational load is reduced in n-times the resources of the entire software Web-system and allows to significantly reduce the cost of redirection of client requests from one node to another. At the next technological stage, the user request passes to the computing node for processing. The architecture of the computing node for processing user's information requests should be organized in a hierarchy style. At the bottom of this hierarchy are important computing modules for processing user requests, the failure of which can temporarily slow down the overall operation of the Web-system, but not paralyze it as a whole. If the lower computation modules do not work, the cloud network Web-system allows to use the corresponding computation modules of other nodes. However, at such organization of computing process it is clear that the level of communication between nodes on processing of user requests should be perfect. Features of the organization of the architecture of the computing node for processing high-user requests requires a clear hierarchy of these modules. Its essence is that each of the computing modules of the node must be as isolated from the others as possible and to exchange information at the level of common system messages while being information dependent. So, the design of software Web-system should be

developed according to the following principle: «where the failure is a certain part of the Web-services work, which should be controlled at the level of system messages exchange».
The architecture of the hierarchy of failures consists in the fact that any Web-system consists of computing modules, and the next has a sequence of actions when the modules of descendants refused. The information flow in this case is in ascending order, because only the parent computing module contains data about what actions should be performed if one or more modules do not work on the cloud computing platform. However, in cases when one of the computing modules of a node fails or is loaded, a request is made to another module, the highest in its hierarchy. It should also be noted that the main computing module may not participate in the development of the client's request, and act as a Web service, which can be in direct contact with other nodes of the cloud network and simultaneously conduct diagnostics of the corresponding modules of the node. After that, the information flows of data generated by the user are developed, and if they cannot be executed, the management decision is made to transfer the information flow from the idle or heavily loaded module, to one which has redundant resources (Figure 1).

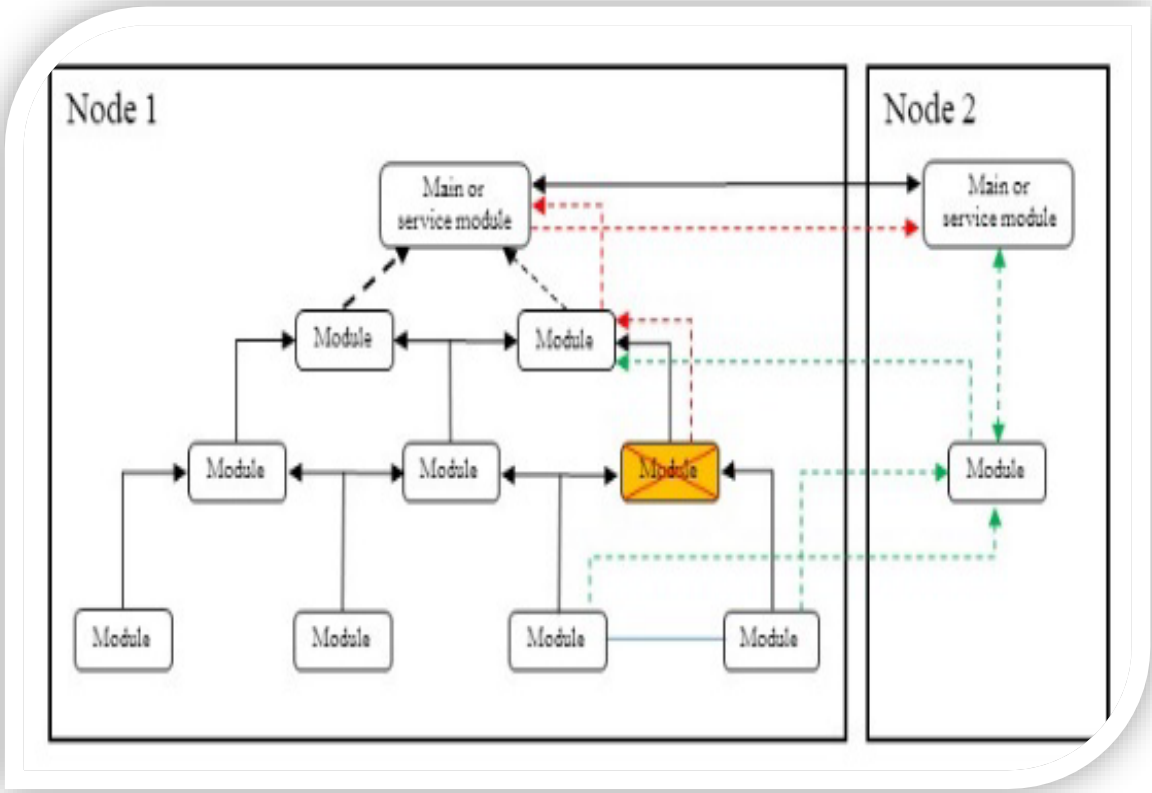

Figure 1: Model of failure hierarchy processing in software Web-systems

The choice of a computing module for executing user requests is based not only on its load, but also on information about the number of failures in other modules of the cloud node. This algorithm is necessary in order not to create information packages of instantaneous data flow for processing from extremely busy node modules to less busy ones. As a result of these manipulations, a complex cloud network for exchange of information arrays of calculations between the corresponding modules of computing nodes is formed. However, the main purpose in building a highly loaded distributed

software Web-systems is to create not such a software system using cloud technology that does not fail during industrial use, and that as long as possible will be able to work and cope with various information challenges and their own mistakes [41]. After the information message flow has been redirected, the computational module does not stop working. So, we can emphasize the strategies that can be implemented in each of these computing modules [15].

In the first strategy we will understand that the computational module was under a significant load. In such a situation, the module executes the whole list of user tasks that came into it or remained in it for processing, performing the primary reboot, and generates an information message to the module that is higher in the hierarchy, and is ready to work. Given that the cloud network of computing nodes modules and the main load balancer constantly exchange information messages about readiness for processing, and the module will receive a new data stream for processing at each subsequent iteration of data flow in the distribution of the load on the nodes.

The snapshot/restore method of data duplication is often used in highly loaded distributed Web systems. Thus, a hybrid system model with a central balancing node of load balancing and cloud network communication between computing nodes allows you to simply add more nodes by applying a cluster structure. High level of communication between computing nodes creates a problem for the data cloning method, because in case of failure of one of the nodes it is necessary to recalculate the integral performance of the Web-system as a whole. It is clear that in this case we can divide the problems connected with the failure of the computational node into two conditional groups. The first group of software and hardware technical problems. Since the redistribution of computational volume occurs at the level of computational modules in case if the software of the developed Web-system generates an error, the system will automatically and quickly restore its performance if its computational module at the top of the hierarchy was properly designed and did not get a cascade error. Then it will alert other computational nodes about errors and redirect the flow of computational information data to other modules. So, in this situation there is no duplication of a particular computational node on an independent platform, but just a launch of a standard instance.

In order to significantly reduce the risk of loss of a significant number of user requests, it is necessary that the balancing computing node contains a buffer that will act as a «retry» in the case when a certain node is physically unavailable. Therefore, using the method of duplication, provides the creation of a typical computing node and simultaneously restores the performance of the Web-system. However, in the development of program systems with the use of cloud technology there is a whole class of distributed algorithms where there are certain structures of information data flows (arrays, records), the size of which depends on the total number of simultaneously interacting processes that perform processing by different computing nodes on independent platforms. An example of such algorithm application is the protocol of making a coordinated decision [34]. When modifying the topology of a distributed highly-advantageous and computational Web-system, the structure of data flows and algorithms of their processing will also change. Therefore, there is an urgent need to modify the scenario of behavior of the used simulation model, namely, to change the algorithms and structures of data flows describing the behavior of the corresponding objects.

4. Implementation of data flow processing methods for recovery of computing nodes

In the development of highly loaded distributed Web-systems, architects make up of several computing nodes, and which are an instance of a single computing system, thus forming a «clustering». However, the modified algorithm of data flow processing requires a clear separation not only on the physical level, but also on the program one. Software of Web-systems is traditionally divided into logical modules, that is by functional purpose [33]. By working out of software Web-system the architect needs its design so that computing modules in the structure had as little logic as

possible as the main task of redistribution of volume of loading is support of system balance. No less important requirement to architecture of software of Web-system is avoidance of strong connectivity. Strong connectivity between computing modules is that one of the modules comes into contact with a large number of other modules on independent platforms. That is, if it breaks down a significant number of modules will send a message to the service module, and that in turn will notify the other nodes of the need for their temporary replacement. To effectively overcome possible computational problems, you can use such methods to restore the functionality of developed software Web-system using cloud technology: the clone_args method stores the values of the arguments that were sent to the computational module that failed [3, 14, 38, 39].

Since the absence of a timely response from the module is classified as its failure in this case, the parent method of the module must get the appropriate information about the arguments that were sent to the child method. However, the direct features of the clone_args method do not end there, in particular there are indirect properties to which they relate:

- the current integral processing time value for the computing module or failure module method;
- the fixed time for which the processing of cloud Web-systems computing nodes failed;
- launch frequency of a computational module, which allows you to determine the need for this module, and therefore, when distributing the load between the nodes need to more effectively allocate the necessary software and hardware resources.

The *report method* allows sending statistical data for the service module about processing the information flows of all child nodes at certain intervals. Also, its use facilitates automated forecasting of degradation dynamics of the developed software Web-system. So, the report method can be used not only to inform about the planned statistical indicators but also when one or several child computing modules have failed.

The *restart method* provides the developed Web-system program with the use of cloud technology by rebooting the computational module in case of its failure. The main purpose of rebooting is to avoid destructive incoming data streams.

The *callback method* allows sending an information message to the parent module about the successful or unsuccessful reboot of the computing node. This method is passed as an argument to the computing module, which should be checked for stable operation after starting or restarting a software Web system that provides a "callback method". Having analyzed the character of errors which can occur in computing modules of the developed program Web-system we'll notice several main reasons: errors in the program code, and also problem data flows and high load in nodes.

The *feedback method* provides communication between the computational nodes of the developed program Web-system using cloud technology and is as independent of others, that's why it can be surrounded by a lot of micro services of Web-systems with which it is necessary to exchange information flows. To do this, of course, you should inform the service computing module, which is on top of the corresponding hierarchy. That's why each computing module implements the feedback method that knows how to contact it and is able to redistribute data flows from modules that fail or are heavily loaded to work nodes. Modified computational model of the recovery method, the modules that have failed, are shown in (Figure 2) [32].

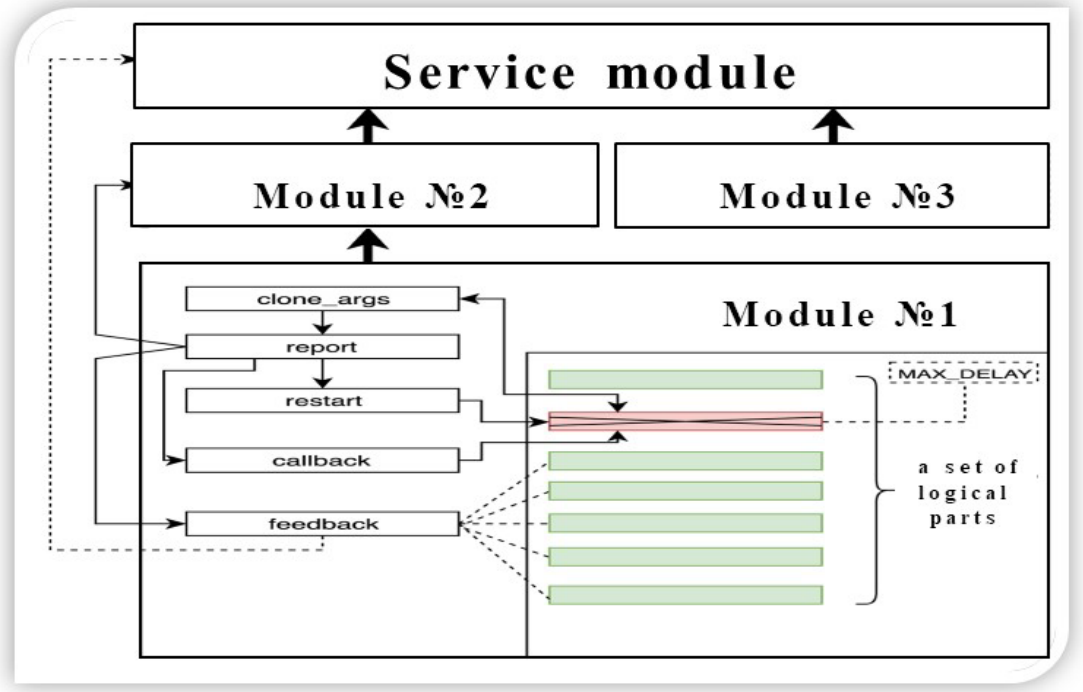

Figure 2: Algorithm of load redistribution in case of failure computing nodes of the software Web-system

This modified algorithm implements only service modules for processing data flows. In contrast to conventional computing modules, service modules provide a much smaller number of operations to obtain a positive result, which directly affects their work. In this case, service modules act as a «stable part of a software system in one computing point» on the node on an independent computing platform. Consequently, when developing a program Web-system it is important to use methods for providing the minimum vulnerability of this computational module. These computational methods constantly inform about the results of work for a certain quantum of time or unplanned in case of leaving the working state. It is clear that the service module has information about the work of all developed software Web-systems. The less time quantum of updating of information streams about results of work of developed program Web-system, the more precisely and effectively it is possible to define necessary computing node for transfer of its load.

Packaging, state transfer and data flow.

As soon as in the developed program Web-system with the use of cloud technology there was a consensus between computing nodes that delegate and accept a part of computational work, it is necessary to prepare a package of raw data for a new node. And only after that to establish a connection between the modules that send user requests to those that fail, and to another working module in another computing node. Also, the service module sends a message marked that the

computing module is damaged – «temporarily unavailable for new requests for processing data flows», then performs the standard procedure of rebooting. This procedure is implemented in each computing module thanks to the corresponding *restart* method, and it is also connected to the service module on the computing node by the *feedback* method. As soon as the software system receives the message that the computational module has failed, *clone_args* copies all the input data streams sent for processing and an indirect set of data about the current state of the system. In this case there is little time for making managerial decisions, so the service module may rather try to pick up a new computational node, simultaneously copying the whole information call thread through the *clone_args* method and directing it to the service module. When analyzing the sequence of the corresponding actions it becomes clear that it will be necessary to form a list of processing calls which are now waiting for other parent computing modules or their methods. However, this list is a kind of queue for processing user requests that are processed by a new working computing module from the created queue. Amazon Simple Queue Service works by the same principle - a service that quickly receives queues of user messages for storage and processing. As soon as the computational node becomes available for information flows of data, the service node composes this data and links the corresponding modules and their states (Figure 3).

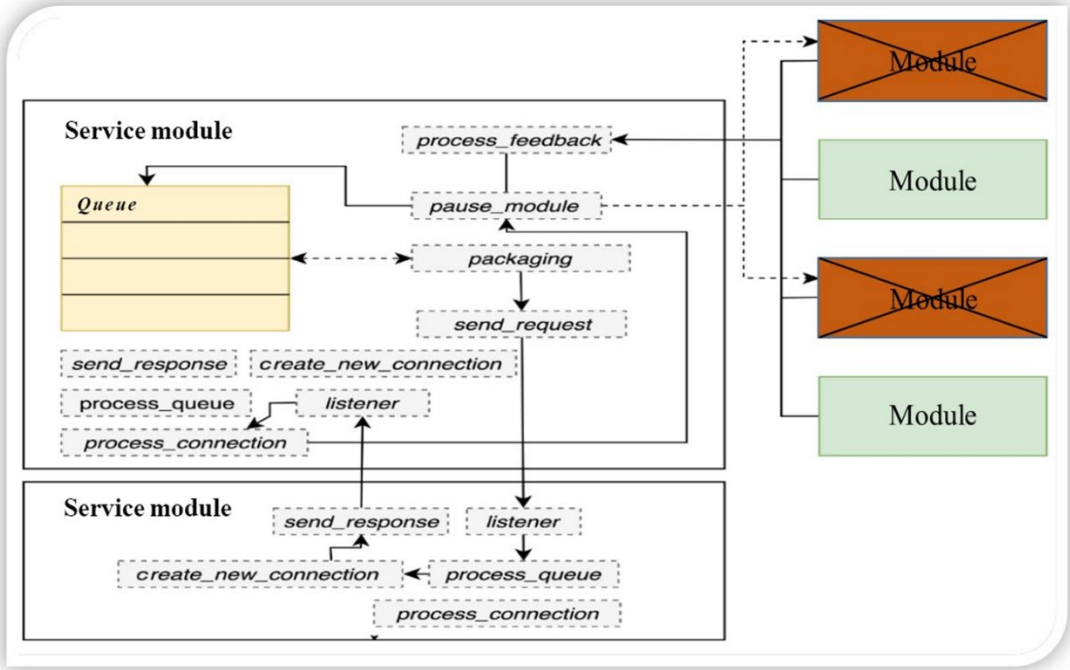

Figure 3: Model of packing, transfer of states and information flows of data

To ensure the reliability of data flow transmission, it is necessary to divide a specific set of these data into appropriate fragments and add to each of them a header with the sequence number. The fragments of information data flows obtained in this way form a segment. On the next step of data processing each segment passes into a package, and then with the help of transport information protocols comes to the recipient's computing node. After the package is delivered to the recipient's computing node, the correctness of receiving the information flow in the segment is checked by means of a checksum recalculation, and it is automatically determined that the previous information flow segments were also successfully received. At this stage, the recipient's computing node sends an information request

to the sender's node about a new data packet or a repeat transmission of the previous data packet. This operation algorithm ensures that all previous packets from the data stream sequence have been successfully received. In the developed modified model computing modules are isolated from each other and do not have information about the general state of the Web-system, while establishing a connection between them, as well as receiving messages about their state. For interaction of computational module, we use asynchronous exchange of information messages where each corresponding module has its own turn of information messages. So, a computing module that sends a message waits for the next notification about its delivery, otherwise the recipient ignores this message.

5. Conclusions

The conducted system analysis examines the basic design principles of distributed risky Web-systems and the technologies that architects of such software systems most often use in their design. We also conclude that redundancy is an indispensable attribute in the design of most risk resilient Web-systems, which are characterized by a high load capacity with respect to user requests. The main criterion in the design of risk resilient Web-systems is their scalability, which is used when operations require significant computational resources, which significantly reduces the performance of the system and requires it to increase its overall capacity. Also, methods for assessing the reliability and resilience of developed risk-resistant software Web-systems are investigated, as any software system must be objectively monitored, and this must be predicted in their work. The main aspects of operation of distributed fault-tolerant software Web-systems are considered, as problems of their administration, as well as restoration of performance in case of failure may arise in the process of highly profitable operation. Analyzing the technologies used in the design of software Web-systems in the study, we have improved methods, according to which a comprehensive analysis of computing load balancing, which is the main task in the design of distributed fault-tolerant software Web-systems. To solve the problem of efficient operation of the developed software system, we use algebraic methods to optimize the balancing of computational load on the nodes, as well as the network model of its distribution, thereby creating a hybrid mechanism of information flows. We consider the theoretical and practical foundations that ensure the effective functioning of the failure hierarchy mechanism, which allows us to provide a risk-sustainable efficient calculation of the computational load of software modules and nodes as a whole. Analyzing a high-loaded distributed fault-tolerant software Web-system, it is found that its fundamental criterion is the efficiency of processing information data streams, generally calculated as the quotient of the sum of all output volumes by the sum of all input data streams. For each specific computational module or node, its value of efficiency is determined. Comparison of the effectiveness of computing nodes is carried out using the linear programming method and at the same time different basic models and their variants. It is proposed to determine the number of computational modules or nodes involved to build a criterion for the boundary of risk efficiency, and for all other cases - the criterion of their inefficiency. Improved functionality of the network model, which differs significantly from the usual sequential and parallel models. The main difference is the ability to aggregate and share significant sets of heterogeneous computing resources for the development of geographically distributed information flows. In many cases this provides significant advantages, since the developed software Web-system requires additional resources that are not available within a single computational node, at the same time they can be obtained from other nodes connected to the functional grid. During the study, we improved the algorithm for transferring the computational load without using redundant resources and proposed a modified model of interaction between computing modules to process information flows within a single node, and the developed software Web-system in general. Also considered the

methodology for determining the causes of failure of computing modules and nodes, highlighted two types of major problems associated with the failure of the node, software and hardware problems. We modify the model of computational process control for the development of information data streams, where modules isolated from each other do not have a common state, but between them you can establish an information connection and receive notifications about their state. Asynchronous exchange of information messages is used for interaction of computing modules, where each module organizes its own message queue. A module sends an information message, waits for an acknowledgement of the corresponding delivery message, but does not receive it if the recipient of the message ignores it. An important aspect of the study is that the risk of stable free computing resources of the developed software Web-system can be found within its capabilities.

References

[1] S.Aleti, S.Bjornander, L.Grunske, I.Meedeniya, ArcheOpterix: An extendable tool for architecture optimization of AADL models, ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software 2009, pp. 73-77

[2] H.Jamalludin, Y.Jamalludin, Analysis of success factors of technology transfer process of the information and communication technology, "International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEES), 2016, Putrajaya, Malaysia, pp. 382-387, doi:10.1109/ICAEES.2016.7888074

[3] V.Andrunyk, A.Vasevych, L.Chyrun, N.Chernovol, N.Antonyu, A.Gozhyj, M.Korobchynskyi, Development of information system for aggregation and ranking of news taking into account the user needs, Paper presented at the CEUR Workshop Proceedings, 2604, 2020, pp. 1127-1171

[4] B.D.Rosenberg, N.Siegel, Critical Quality Factors for Rapid, Scalable, Agile Development, 19-th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Sofia, Bulgaria, 2019, pp. 514-515. doi:10.1109/QRS-C.2019.00101

[5] S.Babichev, V.Lytvynenko, J.Skvor, M.Korobchynskyi, M.Voronenko Information technology of gene expression profiles processing for purpose of gene regulatory networks reconstruction, IEEE 2nd International Conference on Data Stream Mining and Processing, DSMP 2018, 2018, pp. 336-341. doi:10.1109/DSMP.2018.847845

[6] Ch.M.Shoga, B.Boehm, Exploring the Dependency Relationships between Software Qualities, 19-th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Sofia, Bulgaria, 2019, pp. 105-108. doi:10.1109/QRS-C.2019.00032

[7] Zhi et al., Quality Assessment for Large-Scale Industrial Software Systems: Experience Report at Alibaba, 26-th Asia-Pacific Software Engineering Conference (APSEC), Putrajaya, Malaysia, 2019

[8] D.Ageyev, A.Mohsin, T.Radivilova, L.Kirichenko, Infocommunication Networks Design with Self-Similar Traffic, IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), Polyana, Ukraine, 2019, pp. 24-27. doi:10.1109/CADSM.2019.8779314.

[9] D.Ageyev, O.Bondarenko, T.Radivilova, W.Alfroukh, Classification of existing virtualization methods used in telecommunication networks, 9th International Conference on Dependable Systems, Services and Technologies, Kiev, 2018, pp. 83-86. doi:10.1109/DESSERT.2018.8409104.

[10] I.Dronjuk, M.Nazarkevych, O.Fedevych, Asymptotic method of traffic simulations, 2014. doi:10.1007/978-3-319-05209-0_12
[11] I.Dronyuk, M.Nazarkevych, O.Fedevych, Synthesis of Noise-Like Signal Based on Ateb-Functions. In: Vishnevsky V., Kozyrev D, (eds), Distributed Computer and Communication Networks, DCCN 2015, Communications in Computer and Information Science, vol 601, Springer, Cham, 2016. doi:10.1007/978-3-319-30843-2_14
[12] I.Dronyuk, I.Moiseienko, Ja.Greguš, Analysis of Creative Industries Activities in European Union Countries, The International Workshop on Digitalization and Servitization within Factory-Free Economy, (D&SwFFE 2019) November 4-7, 2019, Coimbra, Portugal Procedia Computer Science 160, pp. 479–484
[13] A.AlOmar, M.W.Mkaouer, A.Ouni, M.Kessentini, On the Impact of Refactoring on the Relationship between Quality Attributes and Design Metrics, ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Porto de Galinhas, Recife, Brazil, 2019, pp. 1-11. doi:10.1109/ESEM.2019.8870177
[14] E.Awad, M.W.Caminada, G.Pigozzi, M.Podlaszewski, I.Rahwan, Pareto optimality and strategy-proofness in group argument evaluation, Journal of Logic and Computation, vol. 27, no. 8, 2017, pp. 2581–2609
[15] A.Qasem, A.Qusef, Team Building Activities for Virtual Teams in Jordanian Companies: Vision and Survey, International Conference of Computer Science and Renewable Energies (ICCSRE), Agadir, Morocco, 2019, pp. 1-7. doi:10.1109/ICCSRE.2019.8807677
[16] A.Galkin, R.Umiarov, O.Grigorieva, D.Ageyev, Approaches for Safety-Critical Embedded Systems and Telecommunication Systems Design for Avionics Based on FPGA, IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Ukraine, 2019, pp. 391-396. doi:10.1109/PICST47496.2019.9061421.
[17] L.Junwen, Z.Ziyan, H.Jiakai, The application of quantum communication technology used in electric power information & communication system confidential transmission, 19th International Conference on Advanced Communication Technology, 2017, pp. 305-308
[18] Lv Shu Ping, Tian-Zhenjie, GA's arithmetic on time-cost optimization in architecture engineering, International Conference on Electric Information and Control Engineering, 2011, pp. 3293-3296
[19] M.Brown, Learning Apache Cassandra - Manage Fault Tolerant and Scalable Real-Time Data Mat Brown, Birmingham: Packt Publishing, 2015, 276 p.
[20] M.Kabir, M.Rashed, Multi-level client server network and its performance analysis. Saarbrücken: LAP Lambert Academic Publishing, 2012, 124 p.
[21] M.L.Abbott, M.T.Fisher, The art of scalability: scalable web architecture, processes, and organizations for the modern enterprise, 2nd Edition, Kindle Edition, Boston: Addison-Wesley Professiona, 2015, 618 p.
[22] M.Vladymyrenko, V.Sokolov, V.Buriachok, A.Platonenko, D.Ageyev, Analysis of Implementation Results of the Distributed Access Control System, IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 2019, pp. 1-6. doi:10.1109/PICST47496.2019.9061376
[23] M.Medykovsky, I.Droniuk, M.Nazarkevich, O.Fedevych, Modelling the Pertubation of Traffic Based on Ateb-functions, In: Kwiecień A., Gaj P., Stera P. (eds), Computer Networks, CN 2013, Communications in Computer and Information Science, vol 370, 2013. doi:10.1007/978-3-642-38865-1_5

[24] M.Medykovskyy, M.Pasyeka, N.Pasyeka, O.Turchyn, Scientific research of life cycle perfomance of information technology, 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2017, 1, pp. 425-428. doi:10.1109/STC-CSIT.2017.8098821
[25] O.Mishchuk, R.Tkachenko, I.Izonin, Missing Data Imputation through SGTM Neural-Like Structure for Environmental Monitoring Tasks, Advances in Intelligent Systems and Computing, vol. 938, 2020, pp. 142-151. doi:10.1007/978-3-030-16621-2_13
[26] M.Zeeshan, Z.Mehtab, M.W.Khan, A fast convergence feed-forward automatic gain control algorithm based on RF characterization of Software Defined Radio, International Conference on Advances in Electrical, Electronic and Systems Engineering, Putrajaya, Malaysia, 2016, pp. 100-104. doi:10.1109/ICAEES.2016.7888017
[27] H.Mykhailyshyn, N.Pasyeka, V.Sheketa, M.Pasyeka, O.Kondur, M.Varvaruk, Designing network computing systems for intensive processing of information flows of data, 2021. doi:10.1007/978-3-030-43070-2_18
[28] M.A.J.Idrissi, H.Ramchoun, Y.Ghanou, M.Ettaouil, Genetic algorithm for neural network architecture optimization, 3 International Conference on Logistics Operations Management (GOL), 2016, pp. 1-4
[29] N.Pasieka, V.Sheketa, Y.Romanyshyn, M.Pasieka, U.Domska, A.Struk, Models, methods and algorithms of web system architecture optimization, IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 2019, pp. 147-153. doi:10.1109/PICST47496.2019.9061539.
[30] M.Nazarkevych, M.Logoyda, O.Troyan, Y.Vozniy, Z.Shpak, The ateb-gabor filter for fingerprinting. in international conference on computer science and information technology, 2019, September, Springer, Cham, 2019, pp. 247-255
[31] O.Riznyk, Yu.Kynash, O.Povshuk,V.Kovalyk, Recovery schemes for distributed computing based on bib-schemes, First International Conference on Data Stream Mining & Processing (DSMP), 2016, pp.134-137
[32] P.Haindl, R.Plösch, Towards Continuous Quality: Measuring and Evaluating Feature-Dependent Non-Functional Requirements in DevOps, International Conference on Software Architecture Companion (ICSA-C), Hamburg, Germany, 2019, pp. 91-94. doi:10.1109/ICSA-C.2019.00024
[33] P.Jain, A.Sharma, P.K.Aggarwal, Key Attributes for a Quality Mobile Application, 10-th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 50-54. doi:10.1109/Confluence47617.2020.9058278
[34] M.Pasyeka, V.Sheketa, N.Pasieka, S.Chupakhina, I.Dronyuk, System analysis of caching requests on network computing nodes, 3rd International Conference on Advanced Information and Communications Technologies, AICT2019 - Proceedings, pp. 216-222. doi:10.1109/AIACT.2019.8847909
[35] M.Pasyeka, T.Sviridova, I.Kozak, Mathematical model of adaptive knowledge testing, 5th International Conference on Perspective Technologies and Methods in MEMS Design, MEMSTECH 2009, 2009, pp. 96-97
[36] R.Paul, J.R.Drake, H Liang, Global Virtual Team Performance: The Effect of Coordination Effectiveness, Trust, and Team Cohesion, IEEE Transactions on Professional Communication, Sept. 2016, 2016, vol. 59, no. 3, pp. 186-202. doi:10.1109/TPC.2016.2583319
[37] R.Privman, S.R.Hiltz, Y.Wang, In-Group (Us) versus Out-Group (Them) Dynamics and Effectiveness in Partially Distributed Teams, IEEE Transactions on Professional

Communication, March 2013, vol. 56, no. 1, 2013, pp. 33-49. doi:10.1109/TPC.2012.2237253
[38] O.Riznyk, O.Povshuk, Y.Kynash, M.Nazarkevich, I.Yurchak, Synthesis of non-equidistant location of sensors in sensor network. 14th International Conference on Perspective Technologies and Methods in MEMS Design, MEMSTECH 2018 - Proceedings, 2018, pp. 204-208. doi:10.1109/MEMSTECH.2018.8365734
[39] S.K.Land, The Importance of Deliberate Team Building: A Project-Focused Competence-Based Approach, In IEEE Engineering Management Review, 1 Secondquarter, june 2019, doi vol. 47, no. 2, 2018, pp. 18-22.:10.1109/EMR.2019.2915600
[40] S.Pradhan, V.Nanniyur, P.Melanahalli, M.Palla, S.Chulani, Quality Metrics for Hybrid Software Development Organizations – Case Study, 19-th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Sofia, Bulgaria, 2019, pp. 505-506. doi:10.1109/QRS-C.2019.00097
[41] T.Aslam, T.Rana, M.Batool, A.Naheed, A.Andaleeb, Quality Based Software Architectural Decision Making, International Conference on Communication Technologies (ComTech), Rawalpindi, Pakistan, 2019, pp. 114-119, doi: 10.1109/COMTECH.2019.8737836
[42] T.B.Alakus, R.Das, I.Turkoglu, An Overview of Quality Metrics Used in Estimating Software Faults, International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 2019, pp. 1-6. doi:10.1109/IDAP.2019.8875925
[43] V.Bandyra, A.Malitchuk, M.Pasieka, R.Khrabatyn, Evaluation of quality of backup copy systems data in telecommunication systems, IEEE International Scientific and Practical Conference Problems of Infocommunications Science and Technology, PIC S&T′2019, 08-11 October 2019, Ukraine, 2019, pp. 329-325. doi:10.1109/PICST47496.2019.9061379
[44] V.Sheketa, M.Chesanovskyy, L.Poteriailo, V.Pikh, Y.Romanyshyn, M.Pasyeka, Case-based notations for technological problems solving in the knowledge-based environment, IEEE 2019 14th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, vol.1, 2019, pp. 10–15
[45] W.Liu, J.Yang, Y.Song, X.Yu, S.Zhao, Research on Software Quality Evaluation Method Based on Process Evaluation and Test Results, 6-th International Conference on Dependable Systems and Their Applications (DSA), Harbin, China, 2020, pp. 480-483. doi:10.1109/DSA.2019.00077.
[46] Y.Romanyshyn, V.Sheketa, L.Poteriailo, V.Pikh, N.Pasieka, Y.Kalambet, Social-communication web technologies in the higher education as means of knowledge transfer, IEEE 14th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, vol.3 ,2019, pp. 35–39.
[47] Y.-F.Zhang, H.-Y.Duan, Z.-L.Geng, Evolutionary mechanism of frangibility in social consensus system based on negative emotions spread, Complexity, vol. 2017, Article ID 4037049, 2017, 8 p.
[48] Z.Říhová, L.Dostálek, Information Management - the Basis for Fulfillment of People's Information Needs, 2021, 11th International Conference on Advanced Computer Information Technologies (ACIT), 2021, pp. 469-472, doi: 10.1109/ACIT52158.2021.9548386