American Impact Review

Original Research

Chaos Engineering and Situational Modeling for Fault Tolerance in High-Load Distributed Systems

Danil Redko*

Senior Software Engineer, ada CX, Toronto, Ontario, Canada

* Corresponding author: danilredko1996@gmail.com

OPEN ACCESS

Citation:
Danil Redko (2026). Chaos Engineering and Situational Modeling for Fault Tolerance in High-Load Distributed Systems. *Am. Impact Rev.*
10.66308/air.e2026057

Received: May 7, 2026
Accepted: July 2, 2026
Published: July 9, 2026

DOI:
10.66308/air.e2026057

ISSN: 3071-124X

Abstract

Introduction. In highly loaded systems with high competitive ability and a variety of requests, the problems of high fault tolerance and competitiveness are relevant. Chaos-engineering, chaos-testing allow you to control disturbances in the system and manage its fault tolerance, increasing competitiveness. Purpose. The research goals of the study are to analyze the tools and stages of chaos-engineering (chaos-testing) for such systems, the possibilities of social engineering in increasing fault tolerance and building a situational model of chaos-testing of such systems. Methods. The work uses methods of analysis and synthesis of systems, mathematical and situational modeling, and optimization. Results. Main results: 1) analytics (principles, tools, stages, metrics, social engineering) to conduct chaos-engineering, chaos-testing to manage fault tolerance of a highly competitive and highly loaded system; 2) building a situational model of fault tolerance under the conditions of the specified probabilistic distribution of ”noise” and the procedure for its identification. Conclusion. The results of the study will simplify the analysis and prediction of failures of a high-load system, conduct its load testing in conditions of chaos and high competition.

Keywords: high-load system, chaos, testing, fault tolerance

Introduction

Technologies and systems that provide high competitiveness ensure not only product competitiveness, but also infrastructure and technological independence. It is important to create and test systems that allow you to gain competitive advantages in practice, especially digital ones.

The history of the problem under study and related works go back to 2011. Netflix database corruption in 2011 posed a problem the migration of the data center to the AWS cloud (AmazonWebServices). There was a need for fault-tolerant service operation when transferring to the AWS site. Netflix has developed tools:

1. Chaos Monkey, disabling virtual machines randomly in a working environment;
2. Chaos Gorilla, access zone failure simulator;
3. Chaos Kong, zone loss simulator.

Gregory Orzell filed a patent application in 2010 (published in 2012) for a technique of validating the resiliency of networked applications during Netflix’s migration to the cloud. In 2015, there was a failure in cloud resources associated with automatic scaling and ”time between failures”.

These and similar events demonstrated that going to the cloud is not a guarantee of automatic scaling and reliability. They served as the basis for the emergence of chaos-engineering. It is an area of controlled chaos, analogous to the scientific field of ”deterministic chaos”.

Therefore, such categories as ”entropy of thinking”, ”institutional entropy”, and ”entropy management” have appeared in economics, especially when designing high-load systems (Rudometkin 2020) and ensuring migration of categories to microservice architectures (Radostaev and Nikitina 2021).

The global systemic crisis in the world is accompanied by increased competition, uncertainty, risks and fluctuations in market environments. Chaos arises in the environment, which can increase the entropy of market systems and various models of technological innovation (Li and Wang 2019).

The relevance of this topic is growing with the growth of diversity, diversity of failure characteristics and competition in high-load information systems.

Chaos Engineering is based on deterministic, accurate principles of studying the possibilities of reducing risk in distributed complex systems and networks, increasing the efficiency of search queries for high-load applications (Smirnov, Chervyakov and Bychkova 2025) in highly competitive conditions (Oleinik 2025).

The main research problems of the work:

1. analysis of the principles, tools and stages of chaos engineering in relation to high-load systems;
2. analysis of the possibilities of social engineering in increasing the fault tolerance of a high-load system;
3. construction and study of the situational model of chaos-testing of these systems and the procedure for its identification.

Traditional testing is based on known scenarios, but chaos-testing is based on experiments, controlled disturbances and team observations of the system’s response to failures and the search for instability zones. First, a hypothesis is put forward about the stable behavior of the system, a test plan is built, then deviations are introduced (chaos variables are introduced) and the real behavior of the system is investigated, which is compared with the expected behavior.

The predecessor of chaos-engineering is antifragility, introduced by Nassim N. Taleb and combining the abilities of a system of living organisms in response to chaos, stress.

So, chaos-engineering is a discipline that deliberately introduces errors (vulnerabilities) into software systems to increase their fault tolerance and the likelihood of risk situations, for example, in supply chains in highly competitive environments (Nekrasov, Sinitsyna and Lukinykh 2021).

The novelty of the work lies in the comparative analysis of forecast data, taking into account the capabilities of the chaos-engineering toolkit.

Methods

Controlled chaos is used to analyze many practical problems, from geopolitics to ”crowd behavior”.

Basically, these problems are of an evolutionary ”non-linear” nature. They are aimed at maintaining the stability of the evolutionary strategy of a complex system in the environment. The environment affects the system in a chaotic, dynamic and poorly ordered way (at first glance - randomly), with poorly visible, a priori determined connections from the initial conditions and poorly determined divergences and imbalances in the forecasts (results).

Chaos requires new systems research methodology, new research methods and tools. This requires strategic, global challenges (Nuryshev 2025) and the desire for the sustainability of high-load systems. The international environment itself is chaotic, synergistic, with strongly nonlinear processes. It has no sustainable resource provision, goal-setting, or sustainable strategy for evolution globally.

For example, chaos-engineering is effective in ”clouds” and ”fogs”, it minimizes data leaks, configuration or administration errors. In particular, chaos-engineering can inject controlled chaos into the platform by disabling SG (Security Group) rules or otherwise deliberately affecting traffic or other information systems.

Chaos is also used to assess the impact of information noise on decision-making (Vertinova et al 2022).

Results

Analysis of principles, tools and stages of chaos-engineering

Controlled chaos-engineering introduces chaos, starting with non-critical zones, services and gradually increases power and scalability. It responds to failures and risks, reduces them, and manages proactive fault tolerance.

Stages of chaos engineering:

1. formation of a tested and relevant hypothesis about the system response to instability factors, in comparison with a normal, stable environment (for example, ”error rate”, ”time between failures”, ”delay”, etc.);
2. search for metrics of stable system operation and compare them with emergency ones (for example, ”percentage of requests to API”, ”request execution time”, ”error rate”, ”number of unprocessed requests in the queue”);
3. identification of effects and key factors, variables (for example, the effect of ”server crash” with variables ”time”, ”load”, ”flow”, etc.);
4. construction of a cognitive model of chaos dynamics, compliance of its behavior with hypotheses, requirements for stability, scaling, controllability of the ”explosion radius” in the environment;
5. situational chaos experiment in a real environment or introducing chaos into the system (for example, with threshold variables for analyzing the response to the ”best” scenario according to the developer - ”reliable network”, ”unlimited traffic”, ”stable topology”, etc.);
6. assessment of the relevance of the model and the hypotheses, tools and metrics of chaos-engineering used (for example, using the Structured What If Technique, SWIT);
7. formation of design solutions based on a gradual increase in the ”chaos radius” for the project team (for example, according to the Simian Army principle for testing chaos, starting from the ”unimportant” component).

Utilities are used (ChaosMesh on the Kubernetes platform, Gremlin on the basis of AWS), libraries for implementing failures in applications (Chaos Toolkit), pipelines for checking stability and reliability.

The Simian Army toolkit was dismantled in 2018, it included services, AWS tools for recovery, traffic movement, configuration, simulated failure, load balancing, delay modeling, etc.

Based on developed practices (Chandra 2024, Emrah; Akhan and Cagatay 2026), the following levels of maturity of chaos-engineering are distinguished:

1. zero (chaos-experiment is carried out irregularly, on a test bench with a frequency of about once a quarter and finding a couple of obvious problems, without much confidence in the results and without automation);
2. the first (regular experiments, for example, weekly with the identification of weak points);
3. second (automatic and conveyor CI/CD experiments, daily, taking into account all incidents, failures);
4. the third (implementation of scenarios of major incidents with a monthly frequency, automatic recovery of less than 15 minutes and trained personnel).

Chaos engineering does not always have obvious (direct) advantages, but it always has effective advantages in the infrastructure, which includes customers (Glukhova et al. 2022: Basiri et al 2016). These include, in particular, the following advantages:

1. high availability and durability (”fault tolerance”);
2. efficiency in maintenance, reengineering;
3. increase of understanding of system functioning in different modes;
4. the possibility of reducing the team and the duration of testing, provided that there is a group, a chaos engineering team in the organization, etc.

The multi-stage, multi-criteria nature of cyberattacks, together with the possibilities of social engineering, must be taken into account in hypotheses about the evolution of the system infrastructure (Kaziev and Shapsigov 2024). Half of cyberattacks use social engineering, cognitive modeling and vulnerability analysis.

Chaos-engineering allows you to effectively use cognitive testing schemes, for example, in a simple version - from four zones (A, B, C, and D):

1. ”A” - ”everything is known and everything is clear”;
2. ”B” - ”everything is known, but not everything is clear”;
3. ”C” - ”everything is clear, but not everything is known”;
4. ”D” - ”not everything is known and understood.”

In zone ”A” testing can recommend a minimum sufficient set of experiments. In zone ”B”, experiments can be recommended to study the effect of the added new experiment (test) on the stress resistance of the system. In zone ”C” - experiments to identify new stress-resistance factors (unknown solutions). In zone ”D” - experiments with emergency situations, shutdowns and analysis of response to them.

More likely and/or dangerous stress situations should be tested first, including after discussion, for example, using the Delphi method. It is necessary to identify the effect of the experiment on the key factors of the system and process. As a result of such a process, new problems and situations may emerge.

Situational modeling of fault tolerance in conditions of uncertainty

$$u(t) = \int_0^t z(l)\, dl.$$

$$f_{max} = (\langle f_{t_1}(i_j), \ldots, f_{t_e}(i_j) \rangle)$$

$$t_i \in T, l = 1, 2, \ldots, k,$$

$$w_k = \frac{f_k}{f_{max}}.$$

$$H(P_1, P_2, \ldots, P_n) = P_1^n P_2^{n-1} \cdots P_n^1 = \prod_{j=1}^{n} P_j^{n-j+1},$$

$$P_1 \geq P_2 \geq \ldots \geq P_j \geq \ldots \geq P_n \geq 0.$$

$$H(P_1, P_2, \ldots, P_n) \to max$$

$$\sum_{j=1}^{n} P_j = 1.$$

$$L(P) = H(P) + \lambda \left(1 - \sum_{j=1}^{n} p_j\right),$$

$$\frac{\partial L}{\partial p_j} = P_1^n P_2^{n-1} \cdots (n-j+1) P_j^{n-j} \cdots P_n^1 - \lambda = 0$$

$$\lambda = H(P)(n-j+1)\frac{P_j^{n-j}}{P_j^{n-j+1}}.$$

$$\frac{n(n+1)}{2} H(P) P_j = H(P)(n-j+1),$$

$$F(t) = F_0 \prod_{i=1}^{n} \left(\frac{x_i(t) - x_i^{max}}{x_i^{opt} - x_i^{min}}\right)^{\beta_i} \left(\frac{x_i^{max} - x_i(t)}{x_i^{max} - x_i^{opt}}\right)^{\beta_i \frac{x_i^{max} - x_i^{opt}}{x_i^{opt} - x_i^{min}}},$$

$$lnlnF(t) = lnF_0 + \sum_{i=1}^{n} \beta_i \left(ln\left(\frac{x_i(t) - x_i^{max}}{x_i^{opt} - x_i^{min}}\right) + \frac{x_i^{max} - x_i^{opt}}{x_i^{opt} - x_i^{min}} ln\left(\frac{x_i^{max} - x_i(t)}{x_i^{max} - x_i^{opt}}\right)\right).$$

$$\Phi\left(\beta_1, \beta_2, \ldots, \beta_n\right) = \sum_{j=1}^{N} (lnF\left(t_j\right) - lnf_j)^2 \to min,$$

$$\frac{\partial \Phi}{\partial \beta_i} = 0, j = 1, 2, \ldots, n,$$

Situational modeling models are used in the problem we are considering, insufficiently often and efficiently. As a rule, traditional models are used - regression (statistical), Markov, tree-like, etc.

The purpose of situational decision-making on fault tolerance is to update operational information in scenarios for making the best possible decisions, to search for "confidence points" of stakeholders.

Figure 1 shows the structure of the situational modeling system of a highly competitive system.

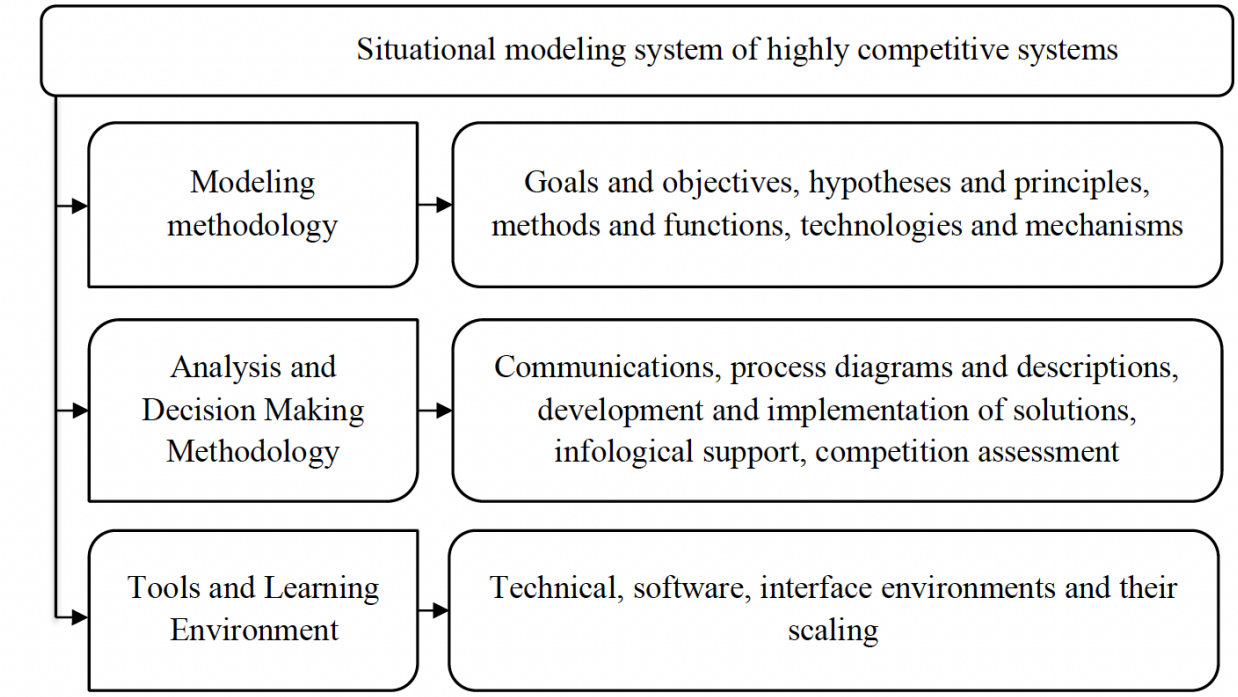

Figure 1. Structure of the situational modeling system (author's diagram)

The situational model of decision making in a competitive environment is based on a computer experiment, assessment of the behavior of competitors, response to possible and permissible changes. As an example of an information model, consider the following model.

We set the flow and source under relatively short-term stability conditions. The flow $u(t)$ is a random distribution with constant expectation. Let $z\left(t\right) > 0$ be formed from the deterministic signal $x(t)$ and the "white" noise $y(t)$: $z\left(t\right) = x\left(t\right) + y(t)$.

The following output is generated by function $z\left(t\right)$:

$$u\left(t\right) = \int_0^t z\left(l\right) dl.$$

For this process, it is important to estimate the limit u^* and the time T of its achievement. In asymptotic $(t \to \infty)$ the function $x(t)$ is found by solving the equation $u(\tau) = u^*$, i.e. the event implementation time will be equal to: $S : u(\tau) = u^*$.

A predictive assessment of the situation is important if the probability distribution $P(\tau)$ cannot be obtained.

Situational modeling and forecasting of flow stability allows you to identify risk areas, business prospects. Knowledge of factors is supported by the rules:

1. tracking information flows;
2. adherence to standards;
3. identifying the causes of resource scarcity;
4. internal (external) audit;
5. optimization of payments;
6. reengineering the digital streaming ecosystem;
7. coordination and integration of all the above rules.

Monitoring is required to ensure strategic, for example, financial and economic goals of the company. Let $I = i_1, i_2, \ldots, i_n$, $N = |I|$ - a stream that requires optimization for a certain group of consumer problems. There is a register of operations $K = \langle I_1, I_2, \ldots, I_N \rangle$, where I_i is i -th transaction (procedure) in which the j-th element of the j_i is specified. The application of the stream during the period $T = \langle t_1, t_2, \ldots, t_k \rangle$ will differ, and the frequency of requests (applications) $f_K(i_j)$ will be calculated as:

$$f_{max} = (\langle f_{t_1}(i_j), \ldots, f_{t_e}(i_j) \rangle), t_i \in T, l = 1, 2, \ldots, k,$$

where f_{max} - maximum frequencies, f_{t_l} - frequencies i_j during the t_l period.

Streams can be ranked according to a formalized index:

$$w_k = \frac{f_k}{f_{max}}.$$

The advantages of this ranking are simplified data analysis, model adaptability, accuracy and data filtering.

Forecasting risk situations and flow manageability is an important research approach that helps to identify key problems, risks and resource-reliability of decision making.

In the presence of ”white” noise, uncertainties, it is important to have a relevant measure of uncertainty. For this, the functionality is offered:

$$H(P_1, P_2, \ldots, P_n) = P_1^n P_2^{n-1} \cdots P_n^1 = \prod_{j=1}^{n} P_j^{n-j+1},$$

$$P_1 \geq P_2 \geq \ldots \geq P_j \geq \ldots \geq P_n \geq 0.$$

We formulate the following problem of the conditional maximum of this functionality:

$$H(P_1, P_2, \ldots, P_n) \to max$$

$$\sum_{j=1}^{n} P_j = 1.$$

Using the method of indefinite Lagrange factors, we find:

$$L\left(P\right) = H\left(P\right) + \lambda\left(1 - \sum_{j=1}^{n} p_j\right),$$

from where

$$\frac{\partial L}{\partial p_j} = P_1^n P_2^{n-1} \cdots (n-j+1) P_j^{n-j} \cdots P_n^1 - \lambda = 0$$

or

$$\lambda = H\left(P\right)(n-j+1)\frac{P_j^{n-j}}{P_j^{n-j+1}}.$$

Therefore,

$$\frac{n\left(n+1\right)}{2} H\left(P\right) P_j = H\left(P\right)\left(n-j+1\right),$$

where do we get estimates from $H\left(P\right)$.

Consider another practically oriented model for predicting the probability of failures based on a Cobb-Douglas-Solow function (He and Qing 2023):

$$F(t) = F_0 \prod_{i=1}^{n} \left(\frac{x_i\left(t\right) - x_i^{max}}{x_i^{opt} - x_i^{min}}\right)^{\beta_i} \left(\frac{x_i^{max} - x_i\left(t\right)}{x_i^{max} - x_i^{opt}}\right)^{\beta_i \frac{x_i^{max} - x_i^{opt}}{x_i^{opt} - x_i^{min}}},$$

where *n* is the number of model factors, $x_i\left(t\right)$ is the value of the factor number *i,* $x_i^{max}, x_i^{min}, x_i^{opt}$ is its maximum, minimum and optimal values, β_i is its significance (importance) in the process, *t* is the time of the forecast period, F_0 is the initial value of *F(t).*

Examples of factors taken into account $x_i\left(t\right)$ - the rate of occurrence of vulnerabilities, the load power of the system, etc.

In order to identify the parameters of the β_i model, the following identification procedure is proposed.

Logarithming the function, we obtain:

$$lnlnF\left(t\right) = lnF_0 + \sum_{i=1}^{n} \beta_i \left(ln\left(\frac{x_i\left(t\right) - x_i^{max}}{x_i^{opt} - x_i^{min}}\right) + \frac{x_i^{max} - x_i^{opt}}{x_i^{opt} - x_i^{min}} ln\left(\frac{x_i^{max} - x_i\left(t\right)}{x_i^{max} - x_i^{opt}}\right)\right).$$

From the least squares criterion follows:

$$\Phi\left(\beta_1, \beta_2, \ldots, \beta_n\right) = \sum_{j=1}^{N} \left(lnF\left(t_j\right) - lnf_j\right)^2 \to min,$$

where f_j - monitoring data.

From a system of algebraic equations:

$$\frac{\partial \Phi}{\partial \beta_i} = 0, j = 1, 2, \ldots, n,$$

solving the system, we find $\beta_i, i = 1.2, \ldots, n$, and then, the predicted values *F(t)* for other time values.

Strategies for achieving fault tolerance are ranked and appropriate risk classes can be identified and appropriate fault testing performed for each class (Klimov 2016). This will allow you to implement neural systems, teach them in classes.

Discussion and conclusions

The performed analysis-synthesis makes it possible to recommend ChaosMesh when working with Kubernetes. If for simple experiments, then Gremlin is recommended, and ChaosToolkit is recommended to implement its own chaos scenarios, for example, in Python.

Chaos engineering will help you find vulnerabilities and defects that are not visible under traditional load, in particular, incorrect timeouts.

A chaos experiment without relevant metrics is useless. It complements functional and stress routine testing.

Chaos testing is effective for finding latent effects, situations where server-service interaction bypasses the API gateway. Therefore, it is relevant to continue this study.

In particular, it is of interest to conduct chaos testing of highly competitive systems using administration tools, signature modification and configuration errors, as well as stress resistance testing using parametric modeling settings.

In practice, the results are determined by their applicability by DevOps engineers, in particular, when scaling high-load systems.

References

1. Basiri A. et al. 2016. Chaos Engineering. *IEEE Software.* 3(33): 3541. DOI:10.1109/MS.2016.60
2. Chandra Shekhar Pareek. 2024. Chaos Testing: A Proactive Framework for System Resilience in Distributed Architectures. *International Journal of Science and Research.* 13(11): 851-855. DOI:10.21275/SR241110081650
3. Emrah Esen, Akhan Akbulut, Cagatay Catal. 2026. Chaos experiments in microservice architectures: A systematic literature review. *Computer Standards & Interfaces*. 97(104116). DOI:10.1016/j.csi.2025.104116
4. Glukhova L.V., et al. 2022. Monitoring and manageability of the corporation's digital business. *Bulletin of Volga University named after V.N. Tatishchev*. 2(1): 14-22. DOI:10.51965/20767919_2022_2_1_14

5. He Z, Qing C. 2023. On Solow-Cobb-Douglas Production Function and it's Relation with Marx's Value Theory of Labor-How to Derive Solow-Cobb-Douglas Production Function from Value Theory of Labor. *Political Economy Quarterly.* 2(1): 120-134. https://doi.org/10.26599/PEQ2023.9310106

6. Kaziev V.M., Shapsigov A.H. 2024. Business Infrastructure Resilience to IT Infrastructure Risks and Its Modeling. In: *International Workshop on Advanced Information Security Management and Applications (AISMA-2024). Conference proceedings. Online: 16 October 2024. Springer*. 863: 124-131. DOI:10.1007/978-3-031-72171-7_13

7. Klimov S.M. 2016. Simulation models of testing important information objects in computer attack conditions. *SFU news. Technical sciences.* 8: 27-36. DOI:10.18522/2311-3103-2016-8-2736.

8. Li Y., Wang L. 2019. Chaos in a duopoly model of technological innovation with bounded rationality based on constant conjectural variation. *Chaos, Solution & Fractals*. 120: 116-126. DOI: 10.1016/j.chaos.2018.11.038

9. Nekrasov A.G., Sinitsyna A.S., Lukinykh V.F. 2021. Tools of chaos engineering and organizational resilience in managed supply chains. *Socio-economic and humanitarian journal. Krasnoyarsk. GAU*. 1: 65-77. DOI: 10.36718/2500-1825-2021-1-65-77.

10. Nuryshev G.N. 2025. The doctrine of "controlled chaos" in modern global geopolitics. *URL: http://economics.open-mechanics.com/articles/401.pdf (accessed on: 20.05.2025)*

11. Oleinik N.M. 2025. Theories of competitiveness and competitive advantages in modern realities. *Bulletin of the Siberian Institute of Business and Information Technology.* 14(3): 103-108. DOI: 10.24412/2225-8264- 2025-3-995

12. Radostaev D.K., Nikitina E.Yu. 2021. Strategy of program code migration from monolithic architecture to microservices. *Bulletin of Perm University. Mathematics. Mechanics. Computer science*. 2(53): 65-68. DOI: 10.17072/1993-0550-2021-2-65-68

13. Rudometkin V.A. 2020. Design of high-load systems. *Proceedings of the ISP RAS.* 2020; 32(6): 79-86. DOI: 10.18255/1818-1015-2019-1-101-121

14. Smirnov N.A., Chervyakov L.M., Bychkova N.A. 2025. Improving the efficiency of search queries for high-load applications. *News of Tula State University. Technical sciences*. 2:152-158. DOI: 10.24412/2071-6168-2025-2-152-153

15. Vertinova A.A. et al. 2022. Assessment of the impact of information noise on decision-making. *Leadership and management.* 9(3): 877-890. DOI:10.18334/lim.9.3.116218