UDC 004.054 **DOI: 10.30977/BUL.2219-5548.2022.96.0.71**

APPROACHES TO WEB APPLICATION PERFORMANCE TESTING AND REAL-TIME VISUALIZATION OF RESULTS

Ushakova I., Plokha O., Skorin Yu.
Simon Kuznets Kharkiv National University of Economics

Abstract. Problem. *Today, performance testing is an integral part of the web applications quality assurance whose performance failures and performance issues affect the business of their owners.* ***Goal.*** *The goal of the work is to generalize approaches and methods to improve the quality of web applications and develop recommendations for improving performance testing using open source tools. The object of research is the processes of testing web applications. The subject of research is the approaches, methods and tools of performance testing.* ***Methodology****. The study identified the impact of software performance testing on its quality and its main types, namely load testing, stress testing, volume testing, stability testing. The main stages of performance testing and their content were identified. To implement modern automated testing technologies, the advantages and disadvantages of the most popular tools for testing performance in the modern IT market and continuous visualization of their results were analyzed and identified. The following factors should be considered when selecting a performance testing tool: compatibility, scalability, clarity, and monitoring. Time series databases and visualization tools are used for continuous monitoring of test results together with testing tools.* ***Results.*** *During the practical implementation of the research results, the goals of different types of performance testing, indicators of normal operation of the system without exceeding the permissible limits, test scenarios and test results were identified. Visualization of test results in JMeter is shown and a board for continuous real-time visualization is created.* ***Originality****. The originality of the study lies in unlocking the potential of open source tools for testing the performance of web applications and visualizing its results. On the basis of comparative analysis the spheres of application of tools for performance testing are substantiated.* ***Practical value.*** *The practical value lies in the development of methodological bases for testing the performance of web applications in real time on the example of the connection of tools Jmeter – InfluxDB – Grafana.*

Keywords: *testing, web application, performance, load, software quality, quality metrics.*

Introduction

Nowadays, performance testing is an integral part of web application quality assurance. Performance testing is a set of types of testing, the purpose of which is to determine the efficiency, stability of resource consumption and other attributes of application quality under different loads and usage scenarios [1]. Performance testing tries to find possible vulnerabilities and defects in the system during its development in order to prevent their negative impact on the operation of the program in use.

Website malfunctions and performance issues affect the business of their owners. Thus, in 2021, the largest failure in the history of the Internet was recorded, when Facebook, Instagram and WhatsApp stopped working for several hours. About $ 6,6 billion was lost then, and the company's management had to substantiate itself to users [2]. There are plenty of such examples. The Apple Store lost about $ 25 million in profits due to a 12-hour delay. Delta Airlines canceled about 2,000 flights and suffered $ 150 million in losses due to the failure of the computer system's operations center for 5 hours.

According to Gartner, the average cost of downtime for IT giants is about $ 300,000 per hour of forced inactivity, and in extreme cases can reach $ 540,000 per hour [3]. Companies are losing money, but worst of all, they are losing their business reputation. That is why it is important for businesses to correctly calculate the potential load on their websites both in normal operation and at peak times. Companies turn to performance testing to find out the causes of failures, but it must be done in a timely manner.

About 80 % of users admit that they are unlikely to buy goods or services from a company whose site "hangs". This can be illustrated by the fact that, for example, Pinterest increased site traffic by 15 % without any marketing costs, only speeding it up by 40 %, and the BBC found that they lose 10 % of users for every second of site load speed [4]. These and many similar examples emphasize the need to spend money on performance testing. Performance testing itself can

ensure the stability of a web application and improve its quality, so improving the performance testing process when creating such programs is important.

Analysis of Publications

Analysis of web application testing technologies allows to divide them into groups, namely: functional, aimed at verifying the compliance of functional requirements of the software to its actual characteristics, and non-functional, aimed at verifying properties that do not belong to the functionality of the system. Non-functional properties characterize reliability, performance, ease of use, scalability and security.

Performance testing is one of the key components of non-functional testing [5–11], because it helps to test the behavior of the program in different situations. The system can work effectively with a certain number of concurrent users, but may become inoperable with many additional thousands of users during peak traffic. Performance testing help determine the speed, scalability, and stability of software. There are different types of performance testing that simulate possible user scenarios and record program indicators of behavior

Performance testing does not necessarily reflect defects in the application. It must ensure that the program works properly regardless of fluctuations in network settings, availability and bandwidth or traffic load. This is practically a subset of performance engineering, as a set of measures for software development and improving all life cycle processes of its development, which are aimed at meeting the requirements of productivity [12]. Therefore, the development and implementation of these tests are crucial to ensure the stability of the web application.

Automating performance testing adds significant benefits to improving the quality of web applications. There are various tools used for these purposes, both licensed and open source [12, 13]. Free tools are of particular interest to small and medium-sized enterprises. The advantages of the open source tool Jmeter are emphasized in [6], and Grinder, NeoLoad, LoadRunner in [12].

Purpose and Tasks

The goal of the research is to generalize approaches and methods to improve the quality of web applications and to develop recommendations for improving performance testing using tools for real-time results analysis.

To achieve this goal, it is necessary to disclose the impact of software testing on its quality, determine the metrics and stages of performance testing, the choice of tools and develop recommendations for the performance testing of web applications.

The impact of software testing on quality

Software testing is an integral part of any modern software development methodology. The essence of the software testing process and its importance as a component of software development are revealed in the literature [14]. The role of testing is one of the key in the life cycle model, because it depends on how high quality the product will reach the customer.

Software testing saves development time and defect correction costs because the cost of troubleshooting is proportional to the time it takes to detect it. Fig. 1 shows the impact of software testing costs on the cost of its quality. Decision making to increase or decrease the number of tests can lead to both the detection and omission of many defects. Therefore, determining the optimal number of tests allows you to minimize the time and cost of testing. The figure clearly shows that the optimal cost of testing is when it is equal to the cost of defect correction.

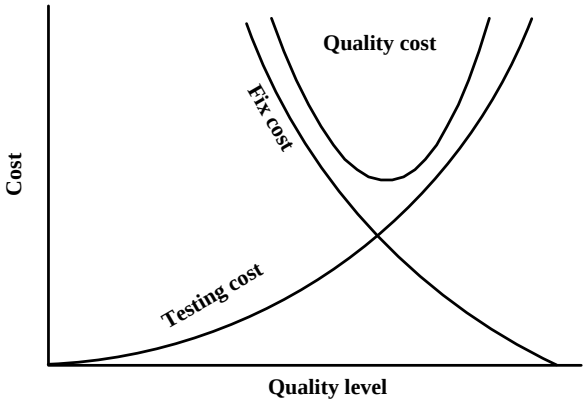

Fig. 1. Dependence of software quality on testing

Recently, the web development market is gaining momentum. And this trend is only growing with each passing year, as entrepreneurs' interest in websites and their mobile versions grows with the transition to the digital economy. As mentioned earlier, an important type of web application testing is performance testing, which includes [1]:

load testing;
stress testing;
volume testing;
stability testing.

Load testing is performed in order to investigate the possibility of the application to keep the

specified quality indicators under load within the specified limits, as well as a certain excess of these limits to determine the margin of safety. Sometimes this type of testing is used as a synonym for "performance testing", but this is not always legitimate, because performance testing is a broader concept.

Stress testing is performed to study the behavior of the program with "abnormal" changes in load in abnormal conditions. It allows you to set limits on the bandwidth of the application, the reliability of the system at extreme or disproportionate loads and answers the73uestionn of the required performance of the system if the current load significantly exceeds the expected maximum.

Volume testing is used to study the performance of the program when processing different amounts of data without increasing the load and operating time.

Stability testing is performed to make sure that the program will withstand the expected load for a long time. During this type of testing, memory consumption is monitored to assess potential losses. Also, this testing allows you to detect performance degradation by reducing the speed of information processing and increasing the response time of the program after prolonged use of the application.

Defining metrics and performance testing stages

For the web application to work successfully, you need to check:

bandwidth, i.e. how fast the server can process requests when a different number of users are connected to the system;

how many simultaneous connections the system can process;

what is the system response time, etc.

The bandwidth of the system includes two components: the number of requests received by the system per second QP_S and the number of responses (transactions) provided by the system per second TP_S:

$$QP_S = {QP}/{T_q},$$

where QP – number of user requests,
T_q – total query execution time;

$$TP_S = {TP}/{T_{tr}},$$

where TP –number of system responses,
T_{tr} – total transaction execution time.

The number of concurrent connections is determined by the number of concurrent users K.

The response time T_r consists of the data transfer time in the network T_{rn} and the processing time T_{ro} (fig. 2):

$$T_r = T_{rn} + T_{ro},$$

where $T_{rn} = N_1 + N_2 + N_3 + N_4$,
$T_{ro} = A_1 + A_2 + A_3$,
A_1, A_3 – server processing time,
A_2 – database processing time.

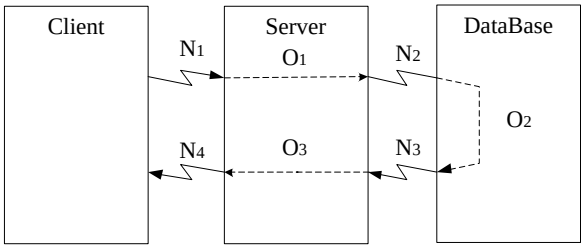

Fig. 2. Components of system response time

Performance testing as a process involves certain stages that need to be clearly defined before testing can begin. Usually distinguish the following stages [14]:

collecting information for testing;
definition of test environment;
testing planning:
creation of tests;
environment configuration;
testing and visualization;
analysis of results and reporting.

The first step in gathering information about web application performance involves determining:

critical functionality to be tested;
expected system response time;
the expected number of users working simultaneously with the system;
expected use of system resources;
future growth of the load on the system, etc.

The definition of a test environment includes the hardware and software and other tools needed to perform the tests. It is necessary that the test environment was as close as possible to the real environment.

Performance testing planning involves actions aimed at defining the main goals of testing and the tasks required to achieve these goals, namely the team of testers, tools, approaches, metrics and priorities. The schedule of performance testing is drawn up at this stage.

The creation of tests includes their design, development of load models, and generation of test data and development of test scenarios in accordance with the project.

The configuration of the environment includes its configuration both on a separate personal computer of the tester for cases of small loading, and a separate usually distributed environment for considerable loading.

Performing stress testing involves running and monitoring test results and usually takes several hours (average two to three hours). If the test is negative, you need to be able to stop the process in real time. In addition, during the experiment, you need to collect the values of many metrics as the server: response time, bandwidth, and the system itself: resource loading, system errors, and so on. To do this, it is convenient to use monitoring systems. They allow you to display all the results in one place and thus quickly notice the relationships between different indicators and decide whether to continue or stop testing.

Collected and analyzed results, identified dependencies are provided to all stakeholders to make further decisions on the quality of the application.

Rationale for choosing testing tools, storing and visualizing results

The following factors should be considered when choosing a performance testing tool:

interoperability,

scalability,

clarity,

monitoring.

Interoperability. Keep in mind that the tools will be used in general in the company, not just for an individual project. Therefore it is necessary to consider in addition the following factors:

protocols used by the system and which of them will be checked;

interfaces to external components such as software components, or possibly to full integration, for example in the CI process;

interoperability with platforms and their versions used to host tools and platforms with which tools interact to monitor and create loads.

Scalability. Performance testing should allow you to track how software is behaving under pressure and provide information on how it can handle scalability.

Clarity should be taken into account given the technical knowledge required by professionals to use the tool. This is often ignored and can lead to unqualified testers setting up tests incorrectly, which in turn can lead to inaccurate results. Some open source testing tools require coding skills. The team must make sure that the tester has the necessary skills, experience and training for testing, which requires complex scenarios and a high level of programming and configuration.

Monitoring is taken into account for its sufficiency. In addition, find out the availability of other monitoring tools available in the environment and which can be used to supplement with this tool. Determine whether monitoring can be correlated with certain operations.

There are many tools on the market for load testing: Kinsta APM, WebLOAD, Apache Jmeter, LoadNinja, Loadero, SmartMeter.io, StormForge, LoadView, NeoLoad, LoadUI Pro, Silk Performer, AppLoader, Gatling, BlazeMeter, Rational Performance Tester, k6, Eggplant, Loadster, Akamai CloudTest, Parasoft Load Test, Locust, Grinder, Loader.io, LoadStorm, SolarWinds, Test Studio, Taurus [13]. Therefore, there is a need to justify their choice. Among the most popular tools that meet the need for performance testing are: Apache Jmeter [15] and Loadrunner [16]. They are the market leaders and are popular among testers and developers of leading IT companies.

Table 1 compares Apache Jmeter and Loadrunner performance testing tools.

The main difference between the Jmeter and LoadRunner tools is the openness of the software and its price. Jmeter is open source software that can be easily downloaded from the official website. LoadRunner software is available as a paid version, and the user must pay for its use.

Table 1 – Comparison of performance testing tools

Specifications	Apache Jmeter	Loadrunner
Cost	Free	- Community Edition free for 50 users, - $0.56 per virtual user per day
Code	Open	Micro-Focus (HP)
Platforms and protocols	Java objects Servlets FTP server queries HTTP SOAP Pearl scripts and other	Web services, .net, J2EE, SAP, Siebel, PeopleSoft, Wireless media and other
User interface	Comfortable simple	Comfortable with complex structure
Functionality	Limited	Powerful
Function settings	Yes	No
Users	Developers, small and medium companies	Medium and large companies

Another difference between benchmarking performance testing tools is supported platforms.

Jmeter can support a variety of platforms, such as Java objects, servlets, FTP servers, database queries, HTTP, SOAP, Pearl scripts, and more. It can be easily run for testing on all mentioned platforms. LoadRunner can support platforms such as web services, .net, J2EE, SAP, Siebel, PeopleSoft, wireless media, and more. And all of these platforms can be used to test performance.

Another difference between Jmeter software and LoadRunner is the user interface. The user interface in Jmeter is user-friendly, but less experienced and has fewer features than LoadRunner. The LoadRunner toolkit is technically more advanced and has more advanced functionality but the user interface structure is more complex compared to Jmeter.

Another difference between Jmeter software and LoadRunner is the configuration of software features. Jmeter is an open source tool, so it provides functionality to customize existing features and modify them as required. LoadRunner is not open source, so you have to use existing functions.

Thus, both tools have their advantages and disadvantages, but they remain market leaders in performance testing. Loadrunner will be the best choice for large businesses and Apache Jmeter for small and medium. Therefore, the Apache Jmeter tool will be used for further research.

Jmeter can save the results of each test to files, but after a long time the number of files will be too large. Also, Jmeter generates an extended report only after the tests are completed, so detailed error logs can only be analyzed after the test is completed. But the development team and any stakeholders need to have this information at all times during testing. This will allow you to track test results such as slow transactions, information about API query errors in real time. This is especially true during long tests, because defects can be detected in the test system at the beginning of testing, which leads to the inexpediency of further testing.

Therefore, the next step is to choose the tools for easier storage of test results and their continuous visualization, which will allow monitoring of test results in real time. Typically, non-relational databases are used to store results, which will write real-time test logs to a spreadsheet, and then these data are visualized by a graphing tool.

In the practice of performance testing usually use one of the following approaches [17, 18]:

- InfluxDB та Grafana;
- Elasticsearch, Logstash та Kibana (ELK).

To choose the tools for storing and visualizing the results, their functional features were analyzed (table 2).

Table 2 – Comparison of functionality

Functionality	ELK	Grafana
Input data formats	++	+
Built-in integrations	+++	+
GUI data streams	++	+
Parsing named	+++	a
Input data processing and enrichment	++	a
Processor templates	++	a
Data visualization	+++	+++
Alert	p	+
Own agents	+	+
Expansion opportunities	+++	+
Documentation	++	++
Installation process	+	+

In the table, the signs "+" indicate the presence of a functional with increasing degree of capabilities, the sign "a" means that this functionality is provided by agents, the sign "p" – paid functionality. The analysis of the considered software solutions revealed a trend on delegation of functionality of primary processing of logs to the local agents that simplifies functionality as it is already visible on Grafana's example. For ELK, there is a trend of gradually reducing free functionality and the emergence of more and more paid. ELK-stack is the most full-featured solution, but the alert is only available in paid versions. Grafana is the simplest and most frivolous solution and is suitable for solving narrow problems related to metric data analysis. Therefore, the approach using free tools was chosen: database - InfluxDB and Grafana visualization tool.

Performance testing with Apache JMeter, InfluxDB and Grafana

The study of load testing processes focused on the need to reduce economic risks and take into account all the necessary business and technical components. It was determined to conduct all key performance tests, the components of which are listed in table 3.

Performance testing metrics and metrics were determined according to web application quality requirements. The requirements are listed in Table 4, and the most important metrics for identifying problems are in table 5.

The test collections were created based on the test scripts created in Apache Jmeter. Also, groups of test users for stress testing under nor-

mal conditions, as well as for stress loads were prepared.

Table 3 – Functional objectives

Type of testing	Objectives
Load	Assessing software response speed Assessing work speed of hardware Measurements on the number of users Defining productivity limits
Stress	Assessing software response speed Assessing hardware speed Assessing the ability to restore the system Assessing the stress loads impact
Volumetric	Assessing the dependence of the system on the size of the processed data Assessing the number of simultaneously working with the system users Assessing the capabilities of data warehouses
Stability	Memory loss assessment Detection of errors related to data collection Assessing the stability of work for a long time

Table 4 – Load testing requirements

Testing option	Value
Active users limit	200
Number of flows	50
Response time	15 sec.
Bandwidth	1 Mbps
Server memory usage	500 megabytes
Number of records read/written to the data warehouse at the same time	1000 records

Table 5 – Load testing metrics

Name	Definition
CPU usage	Determines how much time from the specified interval was spent by the processor on the calculation
Memory usage	Availability of physical memory for processing in the system
Bandwidth	The highest possible data transfer rate in the network
Response time	The time between user queries and application responses
Speed hits	The speed of loading application pages per second
Active sessions	The maximum number of sessions that can be activated at one time

The parameters used to test the normal load on the system are shown in table 6. Expected test result: responses to the queries were received correctly, the response delay is not more than 5 seconds.

Table 6 – Indicators of normal operation of the system without exceeding the permissible limits

№	Value
1	200 users log in to the site at the same time
2	All users change the page from 1 to 10 seconds
3	Users leave a request for a consultation
4	Users periodically create orders

Graphs of queries (fig. 3) and responses to system queries (fig. 4) were obtained after running the test in JMeter. Fig. 3 shows the maximum number of requests processed per second in green and the minimum number in red. Fig. 4 shows the maximum number of system responses per second in yellow and the average in green. Their analysis shows that our system processes an average of 30 requests per second, and the median response to the request is 63 milliseconds, which meets our requirements.

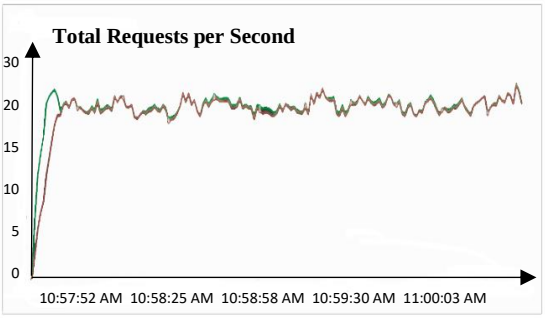

Fig. 3. Number of requests processed per second

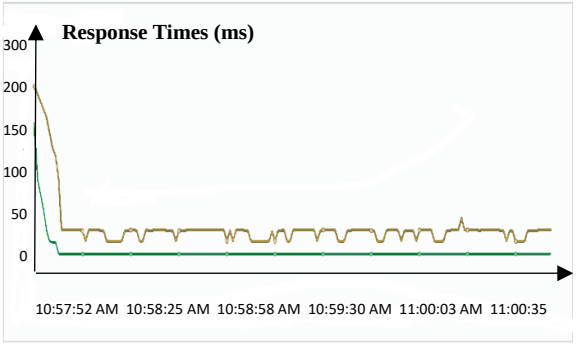

Fig. 4. The number of system responses per-second

A Dashboard was created to visualize the results in Grafana. It allows you to track their dynamics throughout the time of performance testing (fig. 5). There are four graphs on the board

that show the results of system performance testing when processing requests from two sites in Chrome: histogram of the number of requests per second (by status); graphs of the number of requests per second (by instances); latency percentiles of request (green for 99 percentiles, yellow for 50 percentiles, and blue for average); distribution of request latency.

Fig. 5. Visualization of results in Grafana

Modification of indicators occurs with each new pass of test scenarios.

The interactive dashboards helps track performance degradation during testing and identify system vulnerabilities on each of the measured metrics. Analysis of the results of the prepared test scenarios for all types of performance testing makes it possible to determine the readiness of the system for active use, as well as its quality in accordance with the requirements. The results of performance testing according to the developed test scenarios are shown in table 7.

The conducted allows us to draw the following conclusions. The system passed most of the testing scenarios successfully. But checking the time to refresh the application page after prolonged inactive use and checking the execution time of database transactions in stressful situations were not passed.

Table 7 – Testing results

№	Test scenario	Expected result	Yes/No
1	2	3	4
Load testing			
1	Check the operation of the system while the normal number of users are	Work without failures	Yes
2	Check the system operation while a normal number of users are with the authorization page	Work without failures	Yes
3	Check the speed of processing the request to the server with a normal number of requests	Processing speed in an acceptable range	Yes
4	Check the application response time for a normal network connection	The response time is in the acceptable range	Yes
5	Determine the maximum number of users that the program can work with before it shuts down	Maximum number of users when working without failures	Yes
6	Check database execution time when 500 records are read/written at one time	Execution time in an acceptable range	Yes
7	Check database execution time when 1000 records are read/written at one time	Execution time in an acceptable range	Yes
8	Test CPU and memory usage by the application and database server under normal load	Use of resources in the normal range	Yes
9	Check the application response under normal load	Response time in the normal	Yes
10	Check the response time of the application under low load conditions	Response time in the normal range	Yes
11	Check the response time of the application under moderate load	Response time in the normal range	Yes
Stress testing			
12	Check the operation of the system while interacting with an excessive number of users	System shutdown	Yes
13	Check the operation of the system while interacting with an excessive number of users with the authorization page	System shutdown	Yes
14	Check the speed of processing the request to the server in case of excessive number of requests	Requests are processed in turn	Yes
15	Check the response time of the downloaded program with a slow network connection	Response time in the normal range	Yes
16	Check the runtime of the database when 1500 entries are read or written simultaneously	Execution time in an acceptable range	No

1	2	3	4
17	Check CPU and memory usage by program and database server under overload	The system stops working	Yes
18	Check the response time of the program under overload conditions	Response time increases	Yes
Volume testing			
19	Check the system operation when simultaneously downloading files to the warehouse with a nor-mal number of downloads and with a file of normal size	File uploaded successfully	Yes
20	Check the operation of the system when uploading files to the warehouse with an excessive number of downloads and with a file with an excessive volume	File upload declined	Yes
Stability testing			
21	Check for memory loss during peak system interactions	The system is working properly	Yes
22	Check the load time of the application page during peak interaction	File upload declined	Yes
23	Check the refresh time of the application page after an hour of inactive use	The page reloaded successfully without delay	No
24	Check page refresh time during peak user interaction	The page reloaded successfully without delay	Yes

The results of load testing indicate a high level of quality of the software product in terms of performance in accordance with the requirements. Most of the test scenarios were passed with success. These results indicate the readiness of the system for active use at high and stressful loads.

During the testing process, defects that affected system performance and could affect financial losses due to temporary unavailability of the system for customers were identified. Performance and load testing has prevented these risks and identified system vulnerabilities that need to be addressed to further scale the software product.

Conclusion

Digitalization of the business contributes to the further spread of web applications, the quality of which is directly related to performance testing. Performance testing involves researching a system with different loads and monitoring the test results and usually lasts several hours. If testing gives negative results, it should be possible to stop the process in real time to reduce the cost of unnecessary testing.

Many metrics of the system need to be collected during the experiment. The use of a monitoring system during testing allows you to quickly determine the relationships between different indicators and decide whether to continue or stop testing.

An approach to web application performance testing is proposed, which will provide continuous monitoring of performance test results through the use of technologies for storing test results and their visualization. The choice of testing tools was justified, namely Apache Jmeter – for performance testing, InfluxDB – for storing test results, Grafana – for creating dashboards with results. The proposed approach is described by a real example.

References

1. Тестування продуктивності. Qalight. URL: https://qalight.ua/baza-znaniy/testuvannya-produktivnosti (дата звернення: 02.01.2022).
2. Locked out and totally down: Facebook's scramble to fix a massive outage. URL: https://www.theverge.com/2021/10/4/22709575/facebook-outage-instagram-whatsapp (дата звернення: 02.01.2022).
3. The Cost of Downtime. Gartner. URL: https://blogs.gartner.com/andrew-lerner/2014/07/16/the-cost-of-downtime (дата звернення: 02.01.2022).
4. Обновление PageSpeed Insights: что изменилось, на какие метрики обращать внимание? URL: https://siteclinic.ru/blog /technical-aspects/obnovlenie-pagespeed-insights (дата звернення: 02.01.2022).
5. Draheim D., Grundy J., Hosking J. Lutteroth C., Weber G. Realistic Load Testing of Web Applications. *Conference on Software Maintenance and Reengineering (CSMR'06). IEEE Xplore*. 2006. 11 p.
6. Hamza Z. A., Hammad M. Testing Approaches for Web and Mobile Applications: An Overview. *International Journal of Computing and Digital* Systems. 2020. Vol. 9. No. 4. P. 657–664.
7. Israr Gh., Wan M.N., Ahmad M. Web Service Testing Techniques: A Systematic: Literature Review. *International Journal of Advanced Comput-*

er Science and Applications. 2019. Vol. 10. No. 8. P. 443–458.
8. Kao Ch., Lin Ch., Lu H. Toward Automatic Performance Testing for REST-based Web Applications. *ICSEA 2016: The Eleventh International Conference on Software Engineering Advances*. 2016. P. 68–71.
9. Legramante G., Bernardino M., Rodrigues E., Basso F. Systematic Literature Review on Web Performance Testing. *Conference: Escola Regional de Engenharia de Software*. 2020. No. 4. P. 285–295.
10. Legramante G., Bernardino M., Rodrigues E., Basso F. Systematic Literature Review on Web Performance Testing. *2020: Proceedings of the 4th Regional School of Software Engineering. 2020. 11 p.*
11. 10 Best Practices for Application Performance Testing: Leveraging Agile Performance Testing for Web and Mobile Applications. Orasi Software, Inc. 2018. 9 p.
12. Bui S., Shrivastava M., lee E., Dhaliwal J. A case study of testing a web-based application using an open-source testing tool. *Journal of Information Technology Management*. 2015. Vol. XXVI. No. 1. P. 19–30.
13. Top 27 Performance Testing Tools to Use in 2022. URL: https://kinsta.com/blog/performance-testing-tools (дата звернення: 02.01.2022).
14. Crispin L., Gregory J.: Agile testing. Addison-Wesley, 2014. 464 c.
15. Apache JMeter™. URL: https://jmeter.apache. org (дата звернення: 02.01.2022).
16. LoadRunner Professional. URL: https://www. microfocus.com/en-us/products/loadrunner-professional/ overview (дата звернення: 02.01.2022).
17. Grafana. Dashboard anything. Observe everything. URL: https://grafana.com/grafana/
18. What is the ELK Stack? URL: https://www.elastic. co/what-is/elk-stack (дата звернення: 02.01.2022).

References

1. Performance testing. Qalight. Available at: https://qalight.ua/baza-znaniy/testuvannya-produktivnosti (accessed: 02 January 2022).
2. Locked out and totally down: Facebook's scramble to fix a massive outage. 2021. Available at: https://www.theverge.com/2021/10/4. /22709575/facebook-outage-instagram-whatsapp (accessed: 02 Jan. 2022).
3. The Cost of Downtime. Gartner. 2921. Available at: https://blogs.gartner.com/andrew-lerner/2014/07/6 /the-cost-of-downtime (accessed: 02 Jan. 2022).
4. Обновление PageSpeed Insights: что изменилось, на какие метрики обращать внимание? 2019. Accessed 02 Jan. 2022 https://siteclinic.ru/ blog/technical-aspects/ obnovlenie-pagespeed-insights (accessed: 02 Jan. 2022).
5. Draheim D., Grundy J., Hosking J. Lutteroth C., Weber G. Realistic Load Testing of Web Applications. *Conference on Software Maintenance and Reengineering (CSMR'06). IEEE Xplore*, 2006, 11 p.
6. Hamza Z. A., Hammad M. Testing Approaches for Web and Mobile Applications: An Overview. *International Journal of Computing and Digital* Systems, 2020, vol. 9, no. 4, p. 657–664.
7. Israr Gh., Wan M. N., Ahmad M. Web Service Testing Techniques: A Systematic: Literature Review. *International Journal of Advanced Computer Science and Applications*, 2019, vol. 10, no. 8, p. 443–458.
8. Kao Ch., Lin Ch., Lu H. Toward Automatic Performance Testing for REST-based Web Applications. *ICSEA 2016: The Eleventh International Conference on Software Engineering Advances*, 2016, p. 68–71.
9. Legramante G., Bernardino M., Rodrigues E., Basso F. Systematic Literature Review on Web Performance Testing. *Conference: Escola Regional de Engenharia de Softwar*, 2020, no. 4, p. 285–295.
10. Legramante G., Bernardino M., Rodrigues E., Basso F. Systematic Literature Review on Web Per-for mance Testing. *2020: Proceedings of the 4th Regional School of Software Engineering, 2020, 11 p.*
11. 10 Best Practices for Application Performance Testing: Leveraging Agile Performance Testing for Web and Mobile Applications. Orasi Software, Inc. 2018. 9 p.
12. Bui S., Shrivastava M., lee E., Dhaliwal J. A case study of testing a web-based application using an open-source testing tool. *Journal of Information Technology Management*, 2015, vol. XXVI, no. 1, p. 19–30.
13. Top 27 Performance Testing Tools to Use in 2022. Available at: https://kinsta.com/blog /performance-testing-tools (accessed: 02 Jan. 2022).
14. Crispin L., Gregory J. Agile testing. Addison-Wesley, 2014. 464 c.
15. Apache JMeter™. Available at: https://jmeter.apache.org (accessed: 02 Jan. 2022).
16. LoadRunner Professional. Available at: https://www.microfocus.com/en-us/products /loadrunner-professional/overview (accessed: 02 January 2022).
17. Grafana. Dashboard anything. Observe everything. Available at: https://grafana.com/grafana/
18. What is the ELK Stack? Available at: https://www.elastic. co/what-is/elk-stack (accessed: 02 January 2022).

Ushakova Iryna, Ph.D., Assoc. Prof. Informaton System Department, Simon Kuznets Kharkiv National Economic University, tel. +38 066-785-09-92, iryna.ushakova@hneu.net,
Plokha Olena, Ph.D., Assoc. Prof. Informaton System Department, Simon Kuznets Kharkiv National Economic University, tel. +38. 095-570-47-11, badhel@i.ua,
Skorin Yuri, Ph.D., Assoc. Prof. Informaton System Department, Simon Kuznets Kharkiv National

Economic University, tel. +38 066-748-47-51, skorin.yuriy@gmail.com.

Підходи до тестування продуктивності вебзастосунків і візуалізації результатів у реальному часі

Анотація. *Досліджено вплив тестування продуктивності програмного забезпечення на його якість. Для впровадження сучасних технологій автоматизованого тестування були проаналізовані й визначені переваги та недоліки найбільш популярних на сучасному IT-ринку інструментальних засобів тестування продуктивності й візуалізації їхніх результатів, що використовуються для безперервного моніторингу в режимі реального часу. Визначені цілі різних видів тестування продуктивності, показники нормальної роботи системи без перевищення допустимих меж, наведені тестові сценарії та результати тестування. Показана візуалізація результатів тестування в JMeter та створена дошка для безперервної візуалізації в реальному часі.*

Ключові слова: *тестування, вебзастосунок, продуктивність, навантаження, якість програмного забезпечення, метрики якості.*

Ушакова Ірина Олексіївна, к.е.н., доцент кафедри інформаційних систем,
Харківський національний економічний університет ім. С. Кузнеця,
тел. +38 066-785-09-92, iryna.ushakova@hneu.net,
Плоха Олена Борисівна, к.е.н., доцент кафедри інформаційних систем,
Харківський національний економічний університет ім. С. Кузнеця,
тел. +38 095-570-47-11, badhel@i.ua,
Скорін Юрій Іванович, к.т.н., доцент кафедри інформаційних систем,
Харківський національний економічний університет ім. С. Кузнеця,
тел. +38 066-748-47-51, skorin.yuriy@gmail.com.