Scientific Journal of the Ternopil National Technical University
2026, vol. 121, no. 1, pp. 88–98,
ISSN: 2522-4433 Print, ISSN: 2522-4441 Online
https://doi.org/10.33108/visnyk_tntu2026.01.088

Submitted: 2025-11-12 | Revised: 2026-03-05| Accepted: 2026-03-24

Methods for quality assessment of standardized software time indicators

Mykola Kuz [1]*, Yaroslav Bezghachniuk [2], Taras Horodenko [3], Borys Nezamai [4], Andrii Oliinyk [5],

[1] Vasyl Stefanyk Carpathian National University, Ukraine, kuzmykola@ukr.net

[2] Vasyl Stefanyk Carpathian National University, Ukraine, yaroslav.bezghachniuk@pnu.edu.ua

[3] Vasyl Stefanyk Carpathian National University, Ukraine, taras.horodenko.24@pnu.edu.ua

[4] Vasyl Stefanyk Carpathian National University, Ukraine, borys.nezamai@pnu.edu.ua

[5] Vasyl Stefanyk Carpathian National University, Ukraine, andrii.oliinyk@pnu.edu.ua

* Corresponding author kuzmykola@ukr.net

Abstract: *Time parameters of software quality indicators are among the most important when assessing the quality of software products, as they determine performance, stability and resource utilization efficiency. In the article time indicators of software product quality are researched in accordance with international standards ISO/IEC 25022 and ISO/IEC 25023. These standards provide a list of software quality indicators, but do not contain any methods or techniques for determining them. In particular, there are no any measurement techniques for time indicators of software quality. The purpose of this research is study and systematize time indicators of the software products quality and develop methods for their classification and evaluation on a time scale. The feasibility of forming a clear classification of time metrics is substantiated, which covers different time scales – from immediate system reactions to long-term effects of exploitation. The method of structuring indicators into four time ranges (seconds, minutes/hours, days, months), which provides a complex assessment of performance, reliability and stability of software products. An application has been developed that implements the measurement of basic time indicators, in particular, mean response time, median, 95-percentile and dispersion, based on multiple HTTP requests using high-precision timer. The experiment with SLA threshold equals to 150 ms confirms a practical efficiency of the proposed model and it eligibility to industry standards. The obtained results show high performance and stability of the system, that indicates research theoretical statements is relevant. The research emphasizes, that time indicators are a universal indicator of software quality, providing a basis for standardized performance assessment for different types of information system.*

Keywords: *time indicators, software quality, performance, quality assessment.*

1. INTRODUCTION

Statement of the problem. In nowadays researches devoted to software quality assessment, time indicators take a leading place, because they determine performance, stability and resource utilization efficiency. International standards [1, 2] establish set of metrics aimed at quantifying such characteristics as mean response time, mean turnaround time, throughput, system availability and others, which are directly or indirectly related to with time aspect of software functioning, but do not contain methods to determine these characteristics. Methods for time characteristics measurement form the basis for evaluation of software performance and its eligibility to customers requirements. That's why, such methods development is an actual task.

Analysis of recent researches. The significant contribution to the development of approaches to the use of time indicators was made by research focused on the integration of standards into industrial systems. In [3], the feasibility of using metrics from [2] in the context of integrating agents with automated systems was analyzed, where the key importance of such parameters as mean response time and throughput for ensuring correct functioning in real time was demonstrated. Other authors emphasize on the importance of practical introduction of metrics, in particular, in [4] difficulties of using [1] in the field of measuring quality in use were researched, focusing on time parameters as the main indicators of the effectiveness of the end user's interaction with the software product, and recommendations were proposed to simplify the process of their evaluation. This indicates an active movement of the scientific community towards adapting standards to specific applied tasks.

The direction of research related to the performance of cloud systems, where time indicators are decisive, deserves special attention. In [5], it was shown that low-level metrics, such as response time and resource load, can be directly correlated with user's subjective perceptions of performance, which confirms the relevance of the indicators from [2] for monitoring SLA in large-scale distributed environments. This emphasizes a practical value of time characteristics, since they are universal quality markers in different contexts – from local applications to cloud computing.

In turn, in [6] WSQF was developed as a complex quality assessment methodology that specifies the provisions of the ISO 25000 series of standards and allows to obtain benchmarks for commercial software, including time parameters as key components of performance.

The relevance of time parameters research is confirmed in the papers devoted to specific domains, for example embedded systems. In [7] the operational quality model for embedded software was developed, consistent with [8] and [2], which integrates time metrics for task executions in real time into a performance and reliability indicators system. This demonstrates the universality of time indicators, which are not limited only to general systems, but become especially important under tight time constraints. Finally, systematic review of nowadays literature, carried out in [9], shows that lack of unified approaches to performance measuring is a significant barrier to the development of the industry and implementation of standardized procedures, such as [1] and [2], is a necessary condition for achieving comparability and objectivity of results. Theoretical modeling of time characteristics is given in [10], and the practical application of time indicators of software quality is given in [11–13].

The purpose of this work is the research and systematization of time indicators of software products quality and development of methods for their classification and assessment by time scale.

Formulation of the problem. In scope of development of method for evaluation of software products quality time parameters a need arose to create a strict metric classification system by time scale, because this is what the possibility their comparison and usage in different environments depends on. The versatility of time characteristics allows them to be used both for local software solutions and for distributed systems, but each group needs its own method for data gathering and assessment. Therefore, building a classification model that covers all time scales from instantaneous system reactions to long-term business-oriented effects is a prerequisite for further research and development of methods for measuring the time parameters of software product quality indicators.

2. METHODS

This work presents a graphical diagram of the distribution of time metrics into four main categories: with time range in seconds, in minutes or hours, in days, in months. Visualizing this classification in the form of a histogram (Figure 1) allows not only to display the diversity of

identifiers of each indicator, but also to clearly demonstrate their relationship in the overall structure.

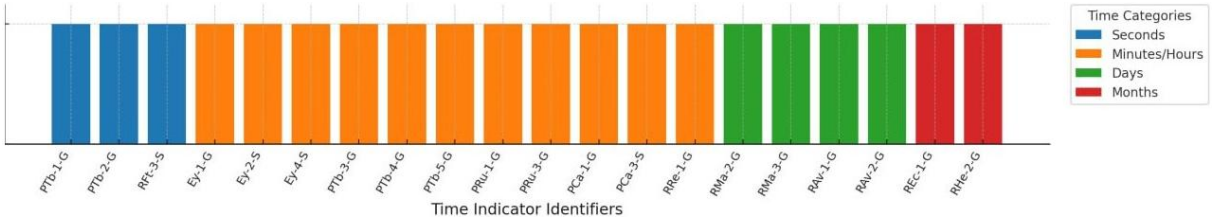

Figure 1. Classification of time indicators of the quality of software products.

In seconds range there are time parameters of the following metrics:

1) Mean response time (ID: Ptb-1-G) – time taken by the by system to respond to user task or system task [2]:

$$\overline{T}_{\text{PTb}-1-\text{G}} = \frac{\sum_{i=1}^{m} T_{\text{PTb}-1-\text{G}} i}{m}, \tag{1}$$

where $T_{\text{PTb}-1-\text{G}} i$ – response time of the system on user task or system task at i-th measurement; m – number of responses measured.

The desynchronization sensitivity is high — an error of 1-2 seconds can distort the result.

2) Response time adequacy (ID: PTb-2-G) – compliance of the system response time with the specified target [2]:

$$X_{\text{PTb}-2\text{-G}} = \frac{\overline{T}_{\text{PTb}-1-\text{G}}}{T_{\text{PTb}-2\text{-G}}}, \tag{2}$$

where $\overline{T}_{\text{PTb}-1-\text{G}}$ – mean response time measured by (1);

$T_{\text{PTb}-2\text{-G}}$ – target response time specified.

The desynchronization sensitivity is high – depends on accuracy of response time measurement.

3) Mean fault notification time (ID: Rft-3-S) – system fault notification speed [2]:

$$\overline{T}_{\text{RFt}-3-\text{S}} = \frac{\sum_{i=1}^{f} \left(Ta_{\text{RFt}-3-\text{S}} i - Tb_{\text{RFt}-3-\text{S}} i\right)}{f}, \tag{3}$$

where $Ta_{\text{RFt}-3-\text{S}} i$ – time at which system fault is reported by system; $Tb_{\text{RFt}-3-\text{S}} i$ – time at which fault is detected; f – number of faults detected.

The desynchronization sensitivity is high — it's crucial to detect moment of fault and moment of system notification in the same time range.

In minutes/hours ranges there are time parameters of the following metrics:

1) Task time (ID: Ey-1-G) — time taken to successfully complete a task [1]:

$$X_{\text{Ey}-1-\text{G}} = T_{\text{Ey}-1-\text{G}}, \tag{4}$$

where $T_{\text{Ey}-1-\text{G}}$ – task time.

The desynchronization sensitivity is middle – the main thing is that the beginning and end of measurements must be detected on the same system.

2) Time efficiency (ID: Ey-2-S) – the efficiency with which users achieve their objectives over time when using the system [1]:

$$X_{\text{Ey}-2-\text{S}} = \frac{A_{\text{Ey}-2-\text{S}}}{T_{\text{Ey}-2-\text{S}}}, \tag{5}$$

where $A_{\text{Ey}-2-\text{S}}$ – number of objectives achieved;

$T_{\text{Ey}-2-\text{S}}$ – time.

The desynchronization sensitivity is moderate — if inrervals are calculated by different systems.

3) Productive time ratio (ID: Ey-4-S) — the proportion of the time that the user is performing productive actions [1]:

$$X_{\text{Ey}-4-\text{S}} = \frac{Ta_{\text{Ey}-4-\text{S}}}{Tb_{\text{Ey}-4-\text{S}}}, \tag{6}$$

where $Tb_{\text{Ey}-4-\text{S}}$ – task time; $Ta_{\text{Ey}-4-\text{S}}$ – productive time:

$$Ta_{\text{Ey}-4-\text{S}} = Ta1_{\text{Ey}-4-\text{S}} + Ta2_{\text{Ey}-4-\text{S}} + Ta3_{\text{Ey}-4-\text{S}} + Ta4_{\text{Ey}-4-\text{S}}, \tag{7}$$

where $Ta1_{\text{Ey}-4-\text{S}}$ – time taken to complete the task;

$Ta2_{\text{Ey}-4-\text{S}}$ – time spent getting help or assistance;

$Ta3_{\text{Ey}-4-\text{S}}$ – time taken recovering from errors;

$Ta4_{\text{Ey}-4-\text{S}}$ – time taken searching ineffectually.

The desynchronization sensitivity is middle – especially if different actions are logged from different devices.

4) Mean turnaround time (ID: PTb-3-G) – time taken for completion of a job or asynchronous process [2]:

$$\overline{T}_{\text{PTb}-3-\text{G}} = \frac{\sum_{i=1}^{k} \left(Tb_{\text{PTb}-3-\text{G}} i - Ta_{\text{PTb}-3-\text{G}} i\right)}{k}, \tag{8}$$

where $Ta_{\text{PTb}-3-\text{G}} i$ – time of starting a i-th job;

$Tb_{\text{PTb}-3-\text{G}} i$ – time of completing the i-th job;

k – number of measurements.

The desynchronization sensitivity is high – starting/completing time should be measured in the same system.

5) Turnarounf time adequacy (ID: PTb-4-G) – turnaround time adequace to the specified targets [2]:

$$X_{\text{PTb}-4-\text{G}} = \frac{\overline{T}_{\text{PTb}-3-\text{G}}}{T_{\text{PTb}-4-\text{G}}}, \tag{9}$$

where $\overline{T}_{\text{PTb}-3-\text{G}}$ – mean turnaround time measured by (8);

$T_{\text{PTb}-4-\text{G}}$ – target turnaround time specified.

The desynchronization sensitivity the same as for mean turnaround time.

6) Mean throughput (ID: PTb-5-G) – mean number of jobs completed per unit time [2]:

$$X_{\text{PTb}-5-G} = \frac{\sum_{i=1}^{z}\left(\frac{A_{\text{PTb}-5-\text{G}}i}{\Delta T_{\text{PTb}-5-\text{G}}i}\right)}{z}, \tag{10}$$

where $A_{\text{PTb}-5-\text{G}}i$ – number of jobs completed during i-th observation time;

$\Delta T_{\text{PTb}-5-\text{G}}i$ – i-th observation time;

z – number of observations.

The desynchronization sensitivity is moderate – the main thing is that calculation of jobs and time are agreed.

7) Mean processor utilization (ID: PRu-1-G) – processor time used to execute a given set of tasks compared to the operation time [2]:

$$X_{\text{PRu}-1-\text{G}} = \frac{\sum_{i=1}^{p}\left(\frac{Tp_{\text{PRu}-1-\text{G}}i}{T_{\text{PRu}-1-\text{G}}i}\right)}{p}, \tag{11}$$

where $Tp_{\text{PRu}-1-\text{G}}i$ – processor time actually used to executea given set of tasks in i-th observation;

$T_{\text{PRu}-1-\text{G}}i$ – operation time to perform the tasks in i-th observation;

p – number of observations.

The desynchronization sensitivity is low – if data taken from one machine.

8) Mean I/O devices utilization (ID: PRu-3-G): – I/O device busy time to perform a given set of tasks compared to the I/O operation time [2]:

$$X_{\text{PRu}-3-\text{G}} = \frac{\sum_{i=1}^{v}\left(\frac{Ta_{\text{PRu}-3-\text{G}}i}{Tb_{\text{PRu}-3-\text{G}}i}\right)}{v}, \tag{12}$$

where $Ta_{\text{PRu}-3-\text{G}}i$ – duration of I/O devices busy time to perform a given set of tasks for i-th observation;

$Tb_{\text{PRu}-3-\text{G}}i$ – duration of I/O operations to perform the tasks for i-th observation;

v – number of observations.

The desynchronization sensitivity is low – when collecting statistics locally.

9) Transaction processing capacity (ID: PCa-1-G) – number of transactions can be processed per unit time [2]:

$$X_{\text{PCa}-1-\text{G}} = \frac{A_{\text{PCa}-1-\text{G}}}{T_{\text{PCa}-1-\text{G}}}, \tag{13}$$

where $A_{\text{PCa}-1-\text{G}}$ – number of transactions completed during observation time;

$T_{\text{PCa}-1-\text{G}}$ – duration of observation.

The desynchronization sensitivity is moderate – counter and timer should be synchronized.

10) User access increase adequacy (ID: PCa-3-S) – number of users can be added per unit time [2]:

$$X_{\text{PCa–3-S}} = \frac{A_{\text{PCa–3-S}}}{T_{\text{PCa–3-S}}}, \tag{14}$$

where $A_{\text{PCa–3-S}}$ – number of users successfully added during observation time;

$T_{\text{PCa–3-S}}$ – duration of observation.

The desynchronization sensitivity is moderate – especially when logging remotely.

11) Mean recovery time (ID: RRe-1-G) – time needed for the software/system to recover from failure [2]:

$$\overline{T}_{\text{RRe–1–G}} = \frac{\sum_{i=1}^{d} T_{\text{RRe–1–G}} i}{d}, \tag{15}$$

where $T_{\text{RRe–1–G}} i$ – total time to recover the downed software/system and re-initiate operation for each failure;

d – number of failures.

The desynchronization sensitivity is high – especially when incidents happen on different nodes.

The next category includes indicators that require somewhat longer observarion or interaction – from several minutes to several hours. Usually these metrics are related to execution of set of tasks , user sessions or mean system load.

In days range there are time parameters of next metrics:

1) Mean time between failure (ID: RMa-2G) – time system/software uptime during their operation [2]:

$$\overline{T}_{\text{RMa–2-G}} = \frac{T_{\text{RMa–2-G}}}{A_{\text{RMa–2-G}}}, \tag{16}$$

where $T_{\text{RMa–2-G}}$ – operation time;

$A_{\text{RMa–2-G}}$ – number of system/software failures actually occured.

The desynchronization sensitivity is low – the main thing is time agreement in logs.

2) Failure rate (ID: RMa-3-G) – average number of failures during a defined period [2]:

$$X_{\text{RMa–3-G}} = \frac{A_{\text{RMa–3-G}}}{T_{\text{RMa–3-G}}}, \tag{17}$$

where $A_{\text{RMa–3-G}}$ – number of failrues detected during observation;

$T_{\text{RMa–3-G}}$ – duration of observation.

The desynchronization sensitivity is moderate – if logs are from different system.

3) System availability (ID: RAv-1-G) – actual proportion of the scheduled system operational time [2]:

$$X_{\text{RAv–1-G}} = \frac{Tf_{\text{RAv–1-G}}}{T_{\text{RAv–1-G}}}, \tag{18}$$

where $Tf_{\text{RAv–1-G}}$ – system operation time actually provided;

$T_{\text{RAv–1-G}}$ – system operation time specified in the operation schedule.

The desynchronization sensitivity is middle – due to the possibility of skewing the availability schedule.

4) Mean down time (ID: RAv-2-G) – time duration, while system stay unavailable when a failure occurs [2]:

$$\bar{T}_{\text{RAv-2-G}} = \frac{T_{\text{RAv-2-G}}}{A_{\text{RAv-2-G}}}, \tag{19}$$

where $T_{\text{RAv-2-G}}$ – total down time; $A_{\text{RAv-2-G}}$ – number of breakdowns observed.

The desynchronization sensitivity is moderate – if breakdowns detected by different nodes.

This group includes indications that require data retrieving over at least several days for reliable assessment. These are mostly reliability and availability metrics, as failures in complex systems can occur rarely and a sufficient period of operation is required to detect them and calculate average characteristics. Also, system availability indicator is considered here, since it usually is assessed in day range.

In month range there are time parameters of next metrics:

1) Return of investment (ID: REc-1-G) – time needed to achieve expected return on investment [1]:

$$X_{\text{REc-1-G}} = T_{\text{REc-1-G}}, \tag{20}$$

where $T_{\text{REc-1-G}}$ – time to achieve return on investment.

The desynchronization sensitivity is high – financial data should be strictly dated.

2) User health and safety impact (ID: Rhe-2-G) – the health and safety impact on users of the software product [1]:

$$X_{\text{RHe-2-G}} = \frac{\sum_{i=1}^{n} \left(Ta_{\text{RHe-2-G}} i * S_{\text{RHe-2-G}} i \right)}{Tb_{\text{RHe-2-G}}}, \tag{21}$$

where n – number of affected people;

$Ta_{\text{RHe-2-G}} i$ – time duration for which the i-th person is affected;

$S_{\text{RHe-2-G}} i$ – degree of significance of the impact on the i-th person;

$Tb_{\text{RHe-2-G}}$ – duration of time from start of system operation.

The desynchronization sensitivity is moderate – especially if data collection is carried out in different departments.

This category includes indicators which represent long-term results of system usage, often going beyond purely technical parameters. It includes business oriented and users aspects, for example return on investment (ROI) – obviously needs months of real usage to compare costs and benefits. Also impact on users health and safety fully appears only with long-term using of the software product. That is why assessment time ranges are months.

3. RESULTS

A software has been developed that implements practical measurement of time indicators of software quality described in standards [1, 2].

Implemented software code uses requests library, which makes multiple HTTP-requests to assessed web application, which is represents real measurements of mean response time of the system. A high precision timer time.perf_counter() is used for each request, which ensures measurement accuracy and minimizes systematic errors impact:

```
start = time.perf_counter()

r = requests.get(url, timeout=10)

end = time.perf_counter()

elapsed_ms = (end - start) * 1000.0
```

The obtained data is accumulated in an array and analyzed using statistical methods, which allows empirically calculate the metrics defined in the article, namely mean response time, median, 95th percentile and variance, which reflects the system stability in the time dimension:

```
avg = np.mean(times)

p95 = np.percentile(times, 95)

var = np.var(times)
```

Besides that, in software percentage of SLA violations is calculated. This calculation reflects how well the actual performance level matches the target user expectations and quality standards:

```
sla_viol = np.mean(np.array(times) > sla_ms) * 100
```

Result of the software run on web site: https://coinmarket.com is presented on Figure 2.

```
Enter site URL: https://coinmarketcap.com/
Number of measurements: 25
SLA threshold (ms): 150

[1] Response time: 104.62 ms, status: 200
[2] Response time: 112.31 ms, status: 200
[3] Response time: 89.71 ms, status: 200
[4] Response time: 101.40 ms, status: 200
[5] Response time: 73.77 ms, status: 200
[6] Response time: 83.23 ms, status: 200
[7] Response time: 70.14 ms, status: 200
[8] Response time: 147.60 ms, status: 200
[9] Response time: 93.52 ms, status: 200
[10] Response time: 103.02 ms, status: 200
[11] Response time: 89.79 ms, status: 200
[12] Response time: 92.39 ms, status: 200
[13] Response time: 98.50 ms, status: 200
[14] Response time: 91.30 ms, status: 200
[15] Response time: 87.91 ms, status: 200
[16] Response time: 93.78 ms, status: 200
[17] Response time: 74.61 ms, status: 200
[18] Response time: 94.27 ms, status: 200
[19] Response time: 91.67 ms, status: 200
[20] Response time: 90.37 ms, status: 200
[21] Response time: 97.56 ms, status: 200
[22] Response time: 121.72 ms, status: 200
[23] Response time: 91.62 ms, status: 200
[24] Response time: 92.27 ms, status: 200
[25] Response time: 152.22 ms, status: 200

--- Results ---
Average time: 95.36 ms
Median: 93.17 ms
95th percentile: 122.64 ms
Variance: 247.39 (ms²)
SLA violations: 0.00 %
```

Figure 2. The result of the software run.

```
Enter site URL: https://stackoverflow.com
Number of measurements: 25
SLA threshold: 150
[1] Response time: 961.13  ms, status: 200
[2] Response time: 634.02  ms, status: 200
[3] Response time: 732.33  ms, status: 200
[4] Response time: 514.17  ms, status: 200
[5] Response time: 747.27  ms, status: 200
[6] Response time: 515.83  ms, status: 200
[7] Response time: 569.01  ms, status: 200
[8] Response time: 517.32  ms, status: 200
[9] Response time: 597.80  ms, status: 200
[10] Response time: 654.21  ms, status: 200
[11] Response time: 806.88  ms, status: 200
[12] Response time: 553.54  ms, status: 200
[13] Response time: 759.47  ms, status: 200
[14] Response time: 560.30  ms, status: 200
[15] Response time: 668.81  ms, status: 200
[16] Response time: 559.27  ms, status: 200
[17] Response time: 558.69  ms, status: 200
[18] Response time: 600.84  ms, status: 200
[19] Response time: 698.19  ms, status: 200
[20] Response time: 718.22  ms, status: 200
[21] Response time: 616.21  ms, status: 200
[22] Response time: 591.65  ms, status: 200
[23] Response time: 588.01  ms, status: 200
[24] Response time: 512.62  ms, status: 200
[25] Response time: 533.43  ms, status: 200

--- Results ---
Average time: 630.77 ms
Median: 597.80 ms
95th percentile: 806.88 ms
Variance: 11540.72 (ms2)
SLA violations: 100.00 %
```

Figure 3. The result of the software run for website with low availability.

In opposite to the result presented on Figure 2 the experiment for another web application was conducted (see Figure 3.) As we can can see on Figure 3, in case of web application with low availability, SLA violations equal to 100% that means that response time for all measurements is greater than SLA threshold value. Also, the variance is much higher than in result from Figure 2. This may be due to an unstable network connection of the tested web application.

4. DISCUSSION

The results from Figure 2, represent a practical usage of time metrics of software quality. 25 measurements of the web application response time were performed during presented experiment, with the SLA threshold set as 150 ms. Such a choice is based on standards of measuring of web application performance and user perception of interface speed.

In accordance with [14, 15] delay up to 100–200 ms is considered practically imperceptible to the user – it is perceived as an instant system response. Similarly, according to AWS CloudWatch engineering standards [16], the 150 ms threshold is often used to test the stability of heavily loaded web services and API, as it allows to separate «fast» responses from those that potentially indicate delays in the network infrastructure.

The results of this experiment show that the system meets the established SLA criteria,

which confirms correctness of the developed quality assessment model. Mean time and median indicate high performance, p95 shows stability of the reaction even in edge cases, and low variance confirms absence of time breakdowns or desynchronisation effect.

Therefore, setting the SLA threshold to 150 ms allowed to test the practical capability of the theoretical model – to show that even a slight excess of time normatives can change the overall quality assessment. The results of the experiment are fully consistent with the analytical conclusions of the article, demonstrating that temporal stability is the determining factor in modern web application performance assessment.

5. CONCLUSIONS

The article summarized the time indicators of the quality of software products according to standards [1, 2]. The developed classification of metrics covers four time ranges – from seconds to months, which makes it possible to adapt the evaluation method to different types of systems. A software was developed to measure the mean response time, which in practice assesses accuracy and stability of time characteristics.

The experimental results showed that mean response time and other parameters remained within the established SLA = 150 ms, which confirms the high performance of the assessed web application. The theoretical classification and practical implementation were consistent each other, proving that time indicators are a determining criterion for evaluating the performance and quality of software products.

Author Contributions: Conceptualization: [Andrii Oliinyk]; Methodology: [Mykola Kuz]; Software: [Taras Horodenko]; Validation: [Yaroslav Bezghachniuk]; Formal analysis: [Borys Nezamai]; Writing – original draft preparation: [Taras Horodenko]; Writing – review and editing: [Yaroslav Bezghachniuk]; Supervision: [Mykola Kuz].

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflicts of Interest: The authors declare no conflict of interest.

Data Availability Statement: All data generated or analyzed during this study are included in this published article.

Declaration of Generative AI and AI-assisted technologies in the writing process: The authors declare that no generative AI or AI-assisted technologies were used in the writing of this manuscript.

REFERENCES

[1] ISO/IEC 25022:2016. Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – Measurement of quality in use. [First edition 2016-06-15]. Switzerland, 2016. 49 p.

[2] ISO/IEC 25023:2016. Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – Measurement of system and software product quality. [First edition 2016-06-15]. Switzerland, 2016. 55 p.

[3] Karnouskos S. et al. The Applicability of ISO/IEC 25023 Measures to the Integration of Agents and Automation Systems, in: 44th Annual Conference of the IEEE Industrial Electronics Society, 2018, pp. 2927–2934. https://doi.org/10.1109/IECON.2018.8592777.

[4] B. Jaradat, G. Weheba, A Practical Guide to ISO 25022: Measurement of Software Quality in Use, Journal of Management & Engineering Integration, 17 (2024), 45–52.

[5] Ravanello A. et al. Associating Performance Measures with Perceived End User Performance: ISO 25023 Compliant Low Level Derived Measures, in: Cloud Computing, GRIDs and Virtualization, 2015, pp. 110–115.

[6] Tsuda N. et al. WSQF: Comprehensive Software Quality Evaluation Framework and Benchmark based on SQuaRE, in: International Conference on Software Engineering, 2019, pp. 61–70. https://doi.org/10.1109/ICSE-SEIP.2019.00045.

[7] Argotti Y. et al. An Operational Quality Model of Embedded Software Aligned with ISO 25000, in: ACM Transactions on Embedded Computing Systems, 2024, pp. 1–41. https://doi.org/10.1145/3691642.

[8] ISO/IEC 25010:2011. Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – System and software quality models. [First edition 2011-03-01]. Switzerland, 2011. 34 p.

[9] Y. Unuchak, T. Unuchak. Approaches to Comprehensive Performance Evaluation of Software Applications: A Systematic Literature Review, in: Human Being, AI & Organization, 2025, pp. 989–1000. https://doi.org/10.18690/um.fov.2.2025.75.

[10] M. Fryz, B. Mlynk, Determination of the characteristic function of discrete-time conditional linear random process and its application, Scientific Journal of TNTU (Tern.), vol. 109, no. 1 (2023), pp. 16–23. https://doi.org/10.33108/visnyk_tntu2023.01.016.

[11] M. Pikuliak, M. Kuz, I. Lazarovych, Y. Kuzyk, V. Skliarov,. Method of Determining the Parameter of Qualitative Evaluation of a Web Forum, Radio Electronics, Computer Science, Control, 3 (2024), pp. 151-159. https://doi.org/10.15588/1607-3274-2024-3-13.

[12] M. Kuz, I. Yaremiy, H. Yaremii, M. Pikuliak, I. Lazarovych, M. Kozlenko, D. Vekeryk, Methods for Evaluating Software Accessibility, Radio Electronics, Computer Science, Control, 3 (2025), pp. 163–172. https://doi.org/10.15588/1607-3274-2025-3-15.

[13] M. V. Kuz, V. M. Andreiko, Evaluation of Quality Characteristics of Software for the Reference Verification Installations for Gasmeters, Ukrainian Metrological Journal, 1 (2018), pp. 43–48. https://doi.org/10.24027/2306-7039.1.2018.134129. [In Ukrainian].

[14] Google Research. Milliseconds Make Millions. Google Cloud Performance Report, 2018. URL: https://services.google.com/fh/files/misc/millisecondsmakemillions.pdf (accessed: 03.10.2025).

[15] Nielsen J. Response Times: The 3 Important Limits. Nielsen Norman Group, 2020. URL: https://www.nngroup.com/articles/response-times-3-important-limits/ (accessed: 03.10.2025).

[16] Amazon Web Services. Monitoring Best Practices: Service Level Objectives. AWS Documentation, 2024. URL: https://docs.aws.amazon.com/ (accessed: 03.10.2025).