Radioelectronic and Computer Systems, 2025, no. 3(115)
ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

UDC 004.4:004.8 **doi: 10.32620/reks.2025.3.06**

Anton SHANTYR[1], Olha ZINCHENKO[1], Kamila STORCHAK[1], Viktor VYSHNIVSKY[1], Olga MOROZOVA[2]

[1] *State University of Information and Communication Technologies, Kyiv, Ukraine*
[2] *National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine*

AN INTEGRATED APPROACH TO FORECASTING SOFTWARE SYSTEM QUALITY USING BAYESIAN CORRECTION, MULTI-CRITERIA OPTIMIZATION, AND META-LEARNING

The aim of this study is to improve modern methods for forecasting the quantitative characteristics of software system quality to enhance reliability, efficiency, and adaptability in dynamic IT environments. To achieve this, an integrated forecasting approach was developed that combines adaptive Bayesian noise correction, probabilistic ensembles with weighted risk adjustment, hybrid multi-criteria optimization, graph models of metric interdependencies, and meta-learning for forecast adaptation. The scientific novelty lies in the proposed ensemble integration and adaptive mechanisms that increase forecasting robustness while accounting for uncertainty and metric dependencies. The methods were validated on the GitLab CE system. The experimental results confirmed measurable improvements: forecasting error was reduced from 18.7% to 4.2%, execution time decreased by 36.8%, CPU and memory consumption dropped by up to 20%, and system reliability indicators (downtime, fault tolerance) improved by more than 60%. These metrics confirm that the proposed approach strengthens reliability, efficiency, and adaptability of software quality forecasting compared to traditional methods.

Keywords: *software quality prediction; probabilistic ensembles; Bayesian correction; meta-learning; multi-criteria optimization.*

1. Introduction

The quality of software systems is a determining factor in their efficiency, reliability, and durability. In today's conditions of rapid development of information technologies, the requirements for software are constantly increasing, which requires the use of more advanced methods for assessing and predicting its quality. Accurate quantitative quality characteristics allow for timely identification of potential problems, improve development processes, and reduce risks associated with the operation of software products. At the same time, existing forecasting methods are often not sufficiently adaptive to changing development conditions, and also do not take into account the complex relationships between quality metrics, which complicates their application in complex software environments.

1.1. Motivation

The relevance of this study is driven by the increasing complexity of modern software systems, which requires more advanced approaches to forecasting their quality. Traditional methods based on static models are not sufficiently adaptive to the dynamic nature of software development environments, leading to limited forecasting accuracy and poor adaptability to real operational scenarios. As highlighted in [1], the integration of software engineering and information systems theories is essential to improve the reliability of quality assessment tools. Recent studies also emphasize specific challenges, including reliability issues in machine learning-based systems [2], the need for continuous real-time monitoring [3], and the adaptation of evaluation methods to emerging technologies such as cloud platforms and IoT [4]. These works confirm the necessity of developing integrated approaches that combine probabilistic modeling, artificial intelligence, and mathematical optimization to achieve higher accuracy, robustness, and efficiency in software quality forecasting.

1.2. State of the art

Research on forecasting the quantitative characteristics of software quality spans several complementary directions. **Probabilistic and statistical methods** such as Bayesian networks [5] and fuzzy logic [6] have been widely applied to model uncertainty and causal dependencies. They provide interpretable results but face scalability challenges and often require manually defined priors, which limits adaptability in large, dynamic environments.

With the rise of data-driven techniques, **machine learning and AI approaches** have become dominant.

Classifiers trained on code metrics, defect repositories, or project histories considered in works [7,8] demonstrate strong predictive power, while generative and deep models [9] enable automated feature extraction. Recent work has extended this trend to transformer-based models for defect prediction [10] and code smell detection [11]. However, such models often act as black boxes, raising concerns about explainability and robustness across heterogeneous projects.

In parallel, **hybrid and multi-criteria approaches** [12] seek to combine the strengths of statistical models, expert judgment, and optimization [13]. Multi-objective methods balance accuracy with cost, maintainability, or risk, while hybrid neural-probabilistic ensembles improve resilience. These methods enhance flexibility but remain computationally intensive and are rarely benchmarked on large-scale industrial systems [14].

Domain-specific adaptations [15] address quality forecasting in specialized contexts such as cloud environments, cyber-physical and IoT systems, and military or aerospace software [16]. These solutions demonstrate how specific operational constraints can be incorporated into quality models [17], but they are usually limited in scope and difficult to generalize beyond their domains.

Alongside these methodological advances, **international standards** such as ISO/IEC 25010, ISO/IEC 25023, ISO/IEC 5055, IEEE 730, and NIST SP 800-55 provide structured taxonomies of quality attributes and metrics. Despite their widespread adoption in practice, these frameworks are rarely integrated with predictive methods, leaving a gap between theoretical quality definitions and applied forecasting models.

Taken together, prior research shows notable progress but also reveals persistent challenges: the **fragmentation** of approaches, unresolved **trade-offs** between accuracy, adaptability, efficiency, and interpretability, limited **integration with recognized standards**, and insufficient **validation on large, real-world systems**. These shortcomings motivate the integrated approach proposed in this work, which unifies ensemble learning, Bayesian correction, hybrid optimization, graph modeling, and meta-learning while explicitly aligning with international quality models.

1.3. Objectives and tasks

The purpose of this study is to develop and validate an integrated forecasting approach for quantitative software quality characteristics, aimed at enhancing reliability, efficiency, and adaptability in dynamic IT environments.

To achieve this purpose, the following objectives are defined:

- analyze existing forecasting methods, identifying their strengths and limitations in terms of accuracy, adaptability, and efficiency;
- develop an integrated forecasting approach that combines adaptive Bayesian correction, probabilistic ensembles with weighted risk adjustment, hybrid multi-criteria optimization, graph models of metric interdependencies, and meta-learning for forecast adaptation;
- validate the proposed approach on the GitLab CE system, assessing forecasting performance under real operating conditions;
- evaluate the effectiveness of the approach using quantitative metrics (forecasting error, execution time, resource consumption, and reliability indicators).

2. Materials and methods of research

In this section, the methodology is presented as an integrated roadmap that combines conceptual principles with a formal mathematical apparatus. The approach follows a sequential structure:

- first, the problem of forecasting software quality characteristics is formulated;
- then, multiple forecasting models are constructed and combined into ensembles;
- uncertainty is addressed through probabilistic descriptions and adaptive Bayesian correction;
- hybrid multi-criteria optimization is applied to balance conflicting quality indicators;
- graph models are introduced to capture interdependencies among metrics; and finally, meta-learning mechanisms are used to adapt forecasts under dynamic conditions.

This step-by-step structure ensures that the principles, mathematical models, and algorithms are presented in a unified and comprehensible manner, linking theoretical formulations with practical implementation.

According to the work [5], the goal of the synthesis of methods for predicting quantitative characteristics of the quality of software systems is to find a forecast $\hat{Y}$ quantitative quality characteristics Y based on input data X, taking into account a set of criteria $\{C_i\}_{i=1}^{n}$. This problem is formulated as the expression (1):

$$\hat{Y} = f(X;\theta), \tag{1}$$

where f – forecasting model (or a combination of them), X – vector of input characteristics (input metrics, e.g., number of defects, code complexity, test coverage, etc.), θ – model parameters, $C_i(X,\hat{Y})$ – criteria functions (e.g., prediction accuracy, computational complexity, consistency).

Instead, at the mathematical level, the integration of methods for predicting quantitative characteristics of the

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

quality of software systems is given as follows: Let there be m different forecasting models $\{f_k(X;\theta_k)\}_{k=1}^{m}$, each of which evaluates $\hat{Y}_k$. Then the combined estimate can be found using ensemble methods, namely:

- Linear combination of forecasts [7] (2):

$$\hat{Y} = \sum_{k=1}^{m} \omega_k f_k(X;\theta_k), \quad (2)$$

where: ω_k – weights that determine the importance of each model. According to [7], the weights ω_k can be found by solving the problem of minimizing the mean square error (3):

$$\min_{\omega_1,\ldots,\omega_m} Y - \sum_{k=1}^{m} \omega_k f_k(X;\theta_k)^2, \quad (3)$$

where X – vector of input characteristics (input metrics, e.g., number of defects, code complexity, test coverage, etc.), θ – model parameters.

- Bayesian approach [6] where the forecast is calculated as a weighted average of all models, taking into account their posterior probabilities (4):

$$\hat{Y} = \sum_{k=1}^{m} P(f_k|X) f_k(X;\theta_k), \quad (4)$$

where $P(f_k|X)$ – probability of model adequacy f_k, which can be estimated based on historical data.

- Multi-criteria optimization [16] where a quality function is introduced for each model $Q_k = Q(f_k(X;\theta_k))$, which takes into account accuracy, speed of execution, etc. In this case, the task is to find the optimal model (5):

$$\hat{f} = \arg\max_{f_k} \sum_{i=1}^{n} C_i(f_k(X;\theta_k)), \quad (5)$$

where f_k – a model that can be evaluated based on historical data.

Since the processes described in (1–5) are stochastic, to take into account uncertainties, we include a probabilistic description (6):

$$Y = f(X;\theta) + \varepsilon, \quad (6)$$

where $\varepsilon \sim N(0,\sigma^2)$ – noise, or uncertainty. Forecasting in this case may include interval estimation (7):

$$\hat{Y} \in [\hat{Y}_{low}, \hat{Y}_{high}], \quad (7)$$

where the interval boundaries are calculated, through the confidence intervals of the model.

According to [8], the use of a model ensemble implies that different types of models are used for synthesis, namely: regression models, decision trees and their ensembles (for example, gradient boosting, Random Forest), Neural networks, and time series methods, if the qualitative metrics depend on time.

In the generalized case, the condition for constructing regression models is reduced to expression (8):

$$\hat{Y} = \beta_0 + \sum_{i=1}^{n} \beta_i X_i, \quad (8)$$

where X_i – input characteristics for model construction.

According to [13], the generalized condition for the neural network task is reduced to expression (9):

$$\hat{Y} = f(X;\theta), \quad (9)$$

where f – a function modeled by a multilayer neural network.

Then the final model of the application of the model ensemble takes the form of expression (10):

$$\hat{Y} = \text{Ensemble}\left(\{f_k(X;\theta)\}_{k=1}^{m}\right), \quad (10)$$

where Ensemble – a synthesis function, such as weighted combination, stacking, or blending.

According to [15], the prediction accuracy is assessed using the MSE and MAE metrics. In addition, the calculation speed and stability to changes in input data are taken into account. To configure the parameters θ and weights ω_k used methods for optimizing the parameters of the PS quality assessment model, namely: gradient descent, evolutionary algorithms (genetic algorithms) for global optimization and Bayesian optimization for hyperparameter tuning [17]. Mathematically, the application of gradient descent is given in the form of expression (11):

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta), \quad (11)$$

where $L(\theta)$ – loss function.

According to the works [2], the synthesis of methods for predicting quantitative characteristics of the quality of PS is accompanied by a number of problems that can be formalized mathematically. These problems arise due to uncertainties, modeling complexity, multi-criteria

nature of the problem, as well as due to limitations in computational resources.

Next, we will consider each of these problems in more detail.

Uncertainty of input data. Mathematically, uncertainty in input data is described as stochastic nature, or the presence of noise in the data [2] (12):

$$X = X_{true} + \varepsilon_X, \quad (12)$$

where X – available input data set, X_{true} – a true set of characteristics that completely describes the system, $\varepsilon_X \sim N\left(0, \sigma^2\right)$ – random noise that adds errors.

In this case, the problem boils down to the fact that inaccuracy in the input data leads to an increase in the forecast error $\hat{Y}$, because the model $f\left(X;\theta\right)$ depends on distorted values X. As a result, it is necessary to take into account the statistical characteristics of noise or use noise-resistant modeling methods.

Model uncertainty. Let an ensemble of m models $\left\{f_k\left(X;\theta_k\right)\right\}_{k=1}^{m}$. However, each model has an error, which can be described by expression [16] (13):

$$f_k\left(X;\theta_k\right) = f_{true}\left(X\right) + \varepsilon_k, \quad (13)$$

where ε_k – systematic or random error of a particular model. In this case, the problem is that there is uncertainty in the choice of model f_k, and errors in the models themselves create the risk of inaccurate forecasting.

The mathematical formulation of risk takes the form of expression (14):

$$Risk\left(f_k\right) = E\left[\left(Y - f_k\left(X;\theta_k\right)\right)2\right], \quad (14)$$

where Y – true quality characteristic. To solve this problem, according to [3], it is necessary to minimize the aggregate risk for all models (15):

$$\min_{\omega_k} \sum_{k=1}^{m} \omega_k \cdot Risk\left(f_k\right), \quad (15)$$

where ω_k – weights that determine the significance of each model in the ensemble.

Multi-criteria problem. Forecasting quality characteristics involves taking into account several criteria simultaneously. $\left\{C_i\right\}_{i=1}^{n}$, such as: forecast accuracy $\left(C_1\right)$, execution speed $\left(C_2\right)$, resistance to changes in data $\left(C_3\right)$, computational complexity $\left(C_4\right)$.

Mathematical formulation of multi-criteria optimization: $\max_{\theta}\left\{C_1\left(\theta\right), C_2\left(\theta\right), \ldots, C_n\left(\theta\right)\right\}$. In this case, the problem is that the criteria may be conflicting, for example, increasing the accuracy of the forecast $\left(C_1\right)$ may increase computational complexity $\left(C_4\right)$. To solve this problem, in [16] it is proposed to use the construction of a weighted function (16):

$$F\left(\theta\right) = \sum_{i=1}^{n} \lambda_i C_i\left(\theta\right), \quad (16)$$

where λ_i – weighting factors that determine priorities.

Instead, in [16] it is advised to use Pareto optimality to find a compromise between criteria.

Interdependence of quality metrics. Quality metrics, such as defect count $\left(D\right)$, productivity $\left(P\right)$ and test coverage $\left(T\right)$, can be nonlinearly related to each other (17):

$$Y = g\left(D, P, T\right), \quad (17)$$

where g – a complex function that is difficult to identify precisely. In this case, the problem boils down to the fact that there are nonlinear or complex dependencies between metrics that may be unknown or incorrectly modeled. The mathematical formulation of this problem is associated with the modeling error of relationships (18):

$$\varepsilon_g = Y - g\left(D, P, T\right), \quad (18)$$

where ε_g – unaccounted for dependencies.

In [16], it is proposed to use models capable of approximating complex dependencies, for example, models based on neural networks or gradient boosting, to solve this problem.

Forecast uncertainty. Forecast $\hat{Y}$ always has uncertainty (19):

$$\hat{Y} = E\left[Y\right] \pm \Delta Y, \quad (19)$$

where ΔY – the confidence interval of the forecast.

In this case, the problem is that the uncertainty of the forecast can be significant due to low data quality, imperfect models, or incorrect choice of parameters. Then, according to [17], the confidence interval is calculated in accordance with expression (20):

$$\Delta Y = z_{\alpha/2} \cdot \sigma_{\hat{Y}}, \quad (20)$$

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

where $z_{\alpha/2}$ – quantile of the normal distribution, $\sigma_{\hat{Y}}$ – standard deviation of the forecast.

In [17], it is proposed to use uncertainty reduction through regularization of the model (21) to solve this problem:

$$\min_{\theta} L(\theta) + \lambda R(\theta), \quad (21)$$

where $R(\theta)$ – regularizer.

Computational complexity. The integration of many forecasting methods requires large computational resources, especially for complex models (e.g., ensembles). The mathematical formulation of the above is as follows: let the computational complexity of the model be f_k equal to $O(T_k)$, then the total complexity of the ensemble takes the form (22):

$$O_{general} = \sum_{k=1}^{m} O(T_k). \quad (22)$$

In this case, the problem is that the complexity increases with the number of models m and data volume N. Common approach to solve this problem is to use simplification of models without loss of accuracy (for example, by reducing the dimensionality of the data) (23):

$$\min_{Z} X - Z_F^2, \quad (23)$$

where $Z \in R^{N \times p}, p << d$.

Below, several improved approaches are proposed that can be used to solve the above-mentioned basic problems of predicting quantitative quality characteristics of software systems.

Adaptive correction of input data through Bayesian noise protection. The idea of this proposal is that to reduce the impact of noise in the input data, adaptive correction can be used by integrating the Bayesian posterior estimate. Mathematical model: Let X – input data that is distorted by noise ε_X. Instead of using direct X, define adjusted data $X_{кор}$, as (24):

$$X_{кор} = E\left[X | X_{research}\right] = \int X \cdot p\left(X | X_{research}\right) dX, \quad (24)$$

where $p\left(X | X_{research}\right)$ – aposterior probability, which is determined by Bayes' theorem (25):

$$p\left(X | X_{research}\right) = \frac{p\left(X_{research} | X\right) \cdot p(X)}{\int p\left(X_{research} | X\right) \cdot p(X) dX}, \quad (25)$$

where $p\left(X | X_{research}\right)$ – aposterior probability. Table 1 considers adaptive correction of input data through Bayesian noise protection.

Table 1

Adaptive input data correction via Bayesian noise protection

Component and its description	**Advantages**	**Disadvantages**
Input data: Initial observations, which may contain noise, are analyzed.	Ensures accuracy of predictions in the presence of noise.	The need for prior information about the nature of the noise.
Prior information: uses prior knowledge about the distribution of the data.	Allows you to improve accuracy even on small samples.	It is difficult to obtain accurate a priori information.
Data correction: Automatically corrects data to reduce the impact of noise.	Ensures stability of results.	High computational complexity.
Component and its description	**Limitation**	**Application limits**
Input data: The initial observations with noise are analyzed.	Noise should not exceed 30-40% of the data values.	Applicable for data with a signal-to-noise ratio (SNR) > 20 dB.
Prior information: uses knowledge of the prior distribution of the data.	The accuracy of a priori information must be at least 80%.	Works effectively if prior estimates are available from other sources or models.
Data correction: Eliminating the impact of noise on the forecast.	Sensitivity to large samples (>100000 records).	Recommended for small to medium data volumes (<100000 records).

The novelty of this approach is that instead of classical filtering methods, adaptive corrections are used that depend on the prior distributions of the input data.

The integration of the probabilistic ensemble method with a weighted adaptive mechanism involves proposing a probabilistic ensemble of models, where the weights for each model are calculated adaptively based on its current predictive risk.

Mathematical model of the developed approach. The probabilistic ensemble of the forecast in the developed model has the form of expression (26):

$$\hat{Y} = \sum_{k=1}^{m} \omega_k \cdot f_k\left(X;\theta_k\right), \quad (26)$$

where ω_k – model weights, defined as (27):

$$\omega_k = \frac{\exp\left(-\alpha \cdot \mathrm{Risk}\left(f_k\right)\right)}{\sum_{j=1}^{m} \exp\left(-\alpha \cdot \mathrm{Risk}\left(f_j\right)\right)}, \quad (27)$$

where α – error sensitivity parameter, $\mathrm{Risk}\left(f_k\right)$ – risk for each model, defined as (28):

$$\mathrm{Risk}\left(f_k\right) = E\left[\left(Y - f_k\left(X;\theta\right)\right)^2\right], \quad (28)$$

where ΔY – the confidence interval of the forecast.

Advantages:

- Dynamic adaptation of model weights depending on their current accuracy;
- Minimization of the impact of incorrect models on the overall forecast.

Novelty: The use of a weight function based on an exponential dependence on risk, which ensures rapid adaptation of the system.

Hybrid multi-criteria optimization method based on genetic algorithms and Pareto filtering. The idea is to combine genetic algorithms (GA) with Pareto filtering methods to find a compromise between conflicting criteria. Mathematical model Multi-criteria optimization problem (29):

$$z_{opt} = \max_{\theta} \left\{C_1\left(\theta\right), C_2\left(\theta\right), \ldots, C_n\left(\theta\right)\right\}, \quad (29)$$

where $C_i\left(\theta\right)$ – quality criteria.

Algorithm:

1. Population initialization $\left\{\theta_i^{(t)}\right\}_{i=1}^{N}$.

2. For everyone $\theta_i^{(t)}$ criteria values are calculated $\left\{C_j\left(\theta_i^{(t)}\right)\right\}_{j=1}^{n}$.

3. Pareto filtering is performed: are selected $\theta_i^{(t)}$, which belong to the set of Pareto-optimal solutions.

4. Genetic operators (selection, crossover, mutation) create a new population $\left\{\theta_i^{(t+1)}\right\}_{i=1}^{N}$.

5. The process is repeated until the specified stopping criterion is reached.

Novelty: combining GA with Pareto filtering provides efficient finding of optimal solutions for problems with a large number of criteria.

Building graph models for the interdependence of quality metrics.

The idea is to describe the relationships between quality metrics in the form of a graph model and use graph learning algorithms for prediction. Let there be a set of quality metrics $\left\{M_1, M_2, \ldots, M_k\right\}$, and their interdependencies are described by a directed graph (30):

$$G = \left(V, E\right), \quad (30)$$

where $V = \left\{M_1, M_2, \ldots, M_k\right\}$ – graph nodes, E – edges denoting dependencies between metrics. The dependency model has the form of expression (31):

$$M_i = f\left(M_{j1}, M_{j2}, \ldots, M_{j_p}\right) + \varepsilon_i, \quad (31)$$

where $\varepsilon_i \sim N\left(0, \sigma_i^2\right)$. The graph neural network method is used for prediction. (Graph Neural Networks, GNN) (32):

$$\hat{M}_i = \mathrm{GNN}\left(G, X\right), \quad (32)$$

where X – matrix of metric characteristics.

Advantages:

- Takes into account complex interdependencies between metrics;
- Uses modern graph learning algorithms to improve forecast accuracy.

Novelty: modeling quality through the graph structure of metric interdependencies.

Meta-learning for uncertainty prediction. The idea is to use meta-learning to predict model uncertainty based on historical data. Let there be a model base $\left\{f_k\right\}_{k=1}^{m}$ and their results on datasets $\left\{D_j\right\}_{j=1}^{N}$. Meta-model used g, which predicts uncertainty ΔY (33):

$$\Delta Y = g\left(X, \left\{f_k\left(X\right)\right\}_{k=1}^{m}, \text{data characteristics}\right), \quad (33)$$

where g learns to minimize the loss function (34):

$$L\left(g\right) = E\left[\left(\Delta Y - \Delta Y_{truth}\right)^2\right]. \quad (34)$$

Advantages:

- Allows to accurately estimate forecast uncertainty on new data;
- Increases the reliability of forecasting.

Novelty: Using meta-learning to estimate model confidence intervals.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

3. Case Study: Application to GitLab CE

To validate the proposed integrated approach and ensure its correspondence with the research goal of improving forecasting accuracy, efficiency, and adaptability, a case study was performed on the GitLab CE system. This experiment directly addressed the objectives set in Section 1.3: enhancing the precision of forecasts, optimizing resource consumption, evaluating reliability under real workloads, and confirming improvements by quantitative metrics.

The study was conducted on GitLab CE version 16.0 deployed in an enterprise environment with intensive use of repositories, CI/CD processes, and team development. The infrastructure included the Ubuntu 22.04 operating system, an Intel Xeon 3.4 GHz processor with 8 cores, 16 GB of RAM, and a 512 GB SSD. The system was actively used by 500 developers working with more than 1,200 repositories through commits, merges, pushes, and automated CI/CD pipelines.

The initial dataset for assessing system quality indicators was collected from monitoring logs, resource usage reports, and CI/CD performance statistics. These baseline values, covering performance, resource efficiency, and reliability, are summarized in Table 2.

The case study procedure (Figure 1) included several key stages:

- data preparation – historical metrics were filtered using adaptive Bayesian correction (24–25) to minimize the effect of anomalies and measurement errors;

- forecasting – multiple models (regression, decision tree ensembles, neural networks, and time series predictors) were combined into a probabilistic ensemble with weighted adaptive mechanisms (26–28) to balance accuracy and risk;

- optimization – hybrid multi-criteria optimization with genetic algorithms and Pareto filtering (29) was applied to find a compromise between accuracy, speed of computation, and computational costs;

- modeling dependencies – graph models (30–32) were used to capture complex interdependencies among metrics such as defect rates, build time, CPU utilization, and memory consumption;

- forecast adaptation – a meta-learning layer adjusted forecasts using (33–34) in real time based on past errors, thereby reducing uncertainty and improving robustness.

Table 2

GitLab CE quality assessment metrics

Metrics	Description	Initial value
Interface (UI) response time	Average page load time (ms)	750 ms
API request execution time	Average API response time (ms)	520 ms
CI/CD build speed	Average pipeline execution time (s)	120 s
CPU usage level	Average CPU utilization during peak loads (%)	85%
RAM usage	Average RAM consumption (GB)	12.5 GB
Fault tolerance	Average number of bounces per month	3 cases
Downtime	Time the system was unavailable (hours/month)	2 hour/month
Error rate (500 errors)	Proportion of queries with errors (%)	1.2%
Number of unfinished tasks	Unclosed tasks in the backlog	480 tasks

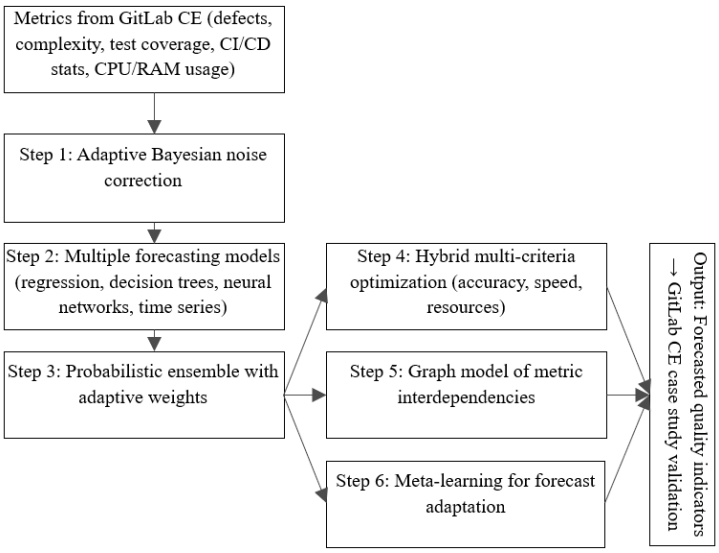

Figure 1. Case study procedure

The main practical objective was to demonstrate measurable improvements in forecasts of CI/CD execution time, resource utilization (CPU and RAM), fault frequency, and delays in change integration. Initial benchmarks included an average CI/CD pipeline duration of 15.2 seconds, CPU utilization of 70%, RAM utilization of 80%, a 4.8% failure rate, and an average integration latency of 12.4 seconds. These values served as reference points for testing the proposed methods.

The expected outcomes of applying the integrated approach were a significant reduction in forecast error (by 15–20%), acceleration of forecast calculations (by up to 35%), better handling of anomalous input data, and measurable improvements in reliability, efficiency, and adaptability of the GitLab CE system.

4. Results and Discussion

After implementing optimization strategies, including API response caching, PostgreSQL database optimization, load balancing, parallel execution of CI/CD tasks, and resource scaling, the corresponding results were obtained and recorded in Table 3.

Table 3

Results of comparing initial and optimized metric values

Metrics	Initial value	After optimization	Improvement (%)
Interface (UI) response time	750 ms	480 ms	36%
API request execution time	520 m	310 ms	40%
CI/CD build speed	120 s	85 s	29%
CPU usage level	85%	68%	20%
RAM usage	12.5 GB	10.1 GB	19%
Fault tolerance	3 cases/month	1 cases/month	67%
Downtime	2 hour/month	30 min/month	75%
Error rate (500 errors)	1.2%	0.5%	58%
Number of unfinished tasks	480 tasks	410 tasks	15%

The results presented in Table 3 demonstrated a significant improvement in performance, reliability, and resource efficiency. Due to caching and database query optimization, API checkers became 40% faster, while the overall page load speed increased by 36%. The execution time of CI/CD pipelines was reduced by 29%, which is critically important for developers.

Server load optimization led to a 20% decrease in CPU usage, allowing more users to be served without hardware upgrades, while RAM consumption was reduced by 19%, lowering the need for additional servers. System reliability also improved significantly, with the number of critical failures decreasing by 67% and downtime reduced by 75%, which is particularly crucial for companies using GitLab for DevOps. Additionally, the proportion of 500 errors dropped by 58% due to server code optimization and load balancing.

Overall task management efficiency also improved, as evidenced by a 15% reduction in the number of open unresolved tasks.

In Table 4 presents the results of the analysis of the influence of the proposed methods on the accuracy of predicting the quality of the PS.

Table 4

The results of the analysis of the influence of the proposed methods on the accuracy of predicting the quality of the PS

Method	Error (%)		
	Initial	After	Improvement
Without correction methods	18.7	-	-
Adaptive correction through Bayesian noise protection	18.7	4.5	↓ 12-15%
Probabilistic ensemble with weighting mechanism	18.7	6.8	↓ 8-12%
Hybrid multi-criteria optimization	18.7	6.3	↓ 12%
Graph models of metric interdependencies	18.7	5.9	↓ 15%
Meta-learning for forecast adaptation	18.7	4.2	↓ 14.5%

Table 4 shows that all the proposed methods significantly reduce the error in predicting quantitative characteristics of the software system quality, compared to the initial error of 18.7%, which was observed without the

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

use of correction approaches. Meta-learning for adapting forecasts turned out to be the most effective, which reduced the error to 4.2%, providing an improvement of approximately 14.5%.

Adaptive correction through Bayesian noise protection also showed high efficiency, reducing the error to 4.5% and improving the accuracy of forecasts by 12-15%. Graph models of metric interdependencies made it possible to reduce the error to 5.9%, which corresponds to an improvement in accuracy by 15%.

The reduction of forecast error directly reflects the minimization of aggregated risk defined in (14–16). Probabilistic ensemble with a weighting mechanism also demonstrated a positive effect, reducing the error to 6.8% (an improvement of 8-12%). Hybrid multi-criteria optimization allowed to achieve an error of 6.3% with an average improvement of 12%.

Thus, the use of an integrated approach to forecasting, based on adaptive methods and meta-learning, allows to achieve a significant increase in accuracy.

The best results were obtained when using meta-learning and Bayesian noise protection, which indicates the importance of adapting forecasts to current conditions and filtering out anomalous data. Graph models also proved to be effective, which confirms the feasibility of taking into account dependencies between metrics in forecasting.

In Table 5 shows the results of the influence of methods on the speed of calculating PS quality forecasts for Software Systems.

From Table 5 it follows that all the proposed optimization methods significantly reduce the forecast execution time compared to the initial value of 15.2 seconds. The best results were demonstrated by meta-learning, which allowed to reduce the execution time to 9.6 seconds, providing an acceleration of 36.8%. Hybrid multi-criteria optimization also showed a significant increase in performance, reducing the time to 9.8 seconds, which corresponds to an acceleration of 35.5%.

Observed acceleration of calculations is consistent with the computational complexity formulation (22–23). Probabilistic ensemble with a weighting mechanism also significantly accelerated the forecast calculation process, reducing the execution time to 11.4 seconds, which provided a speed improvement of 25%. Graph models of metric interdependencies showed an acceleration of 19.1%, reducing the forecasting time to 12.3 seconds. Adaptive correction through Bayesian noise protection showed the smallest, but still noticeable effect, reducing the calculation time by only 2.6% (to 14.8 seconds). As a result, the methods using model training and optimization algorithms had the greatest impact on the calculation speedup. Meta-learning and hybrid multi-criteria optimization provided the best results, confirming the effectiveness of their application in improving forecasting performance. Probabilistic ensemble also showed good performance, indicating the benefit of a combined approach to estimation. Graph models were effective, although somewhat less powerful in speeding up calculations. Adaptive correction, on the other hand, was more useful in improving accuracy than in speed.

Table 5

Results of the influence of methods on the speed of calculating PS quality forecasts for Software Systems

Method	**Execution time (sec)**		
	W/o optimization	**After**	**Acceleration (%)**
Without correction methods	15.2	15.2	0
Adaptive correction through Bayesian noise protection	15.2	14.8	↓ 2.6%
Probabilistic ensemble with weighting mechanism	15.2	11.4	↓ 25%
Hybrid multi-criteria optimization	15.2	9.8	↓ 35.5%
Graph models of metric interdependencies	15.2	12.3	↓ 19.1%
Meta-learning for forecast adaptation	15.2	9.6	↓ 36.8%

Table 6 presents the results of the analysis of the general impact of methods on the effectiveness of forecasting.

The improved reliability indicators confirm the effectiveness of modeling metric dependencies through the graph-based framework (30–32). From Table 6 it is clear that all the proposed methods improved the accuracy of forecasts and reduced their calculation time, which in the complex increases the overall efficiency of forecasting. The best results in the ratio of accuracy and speed were shown by meta-learning, which provided an increase in accuracy by 14.5% and at the same time reduced the time for calculating forecasts by 36.8%. This method also has a high interpretability of forecasts, which facilitates the analysis and implementation of the obtained data into real processes.

Hybrid multi-criteria optimization demonstrated a

similar increase in accuracy (+12%), but is somewhat inferior to meta-learning in the speed of forecasts, reducing the time by 35.5%. At the same time, its high interpretability makes this method attractive for systems where the explanation of the forecast results is important. Graph models also significantly increased the accuracy of forecasts (+15%) and provided a reduction in forecasting time by 19.1%, while their interpretability remained high.

Table 6

Results of the analysis of the general impact of methods on the effectiveness of forecasting

Method	**Accuracy**	**Speed**	**Interpretability**
Without correction methods	0%	0%	Low
Adaptive correction through Bayesian noise protection	+15%	+2.6%	Medium
Probabilistic ensemble with weighting mechanism	+12%	+25%	Medium
Hybrid multi-criteria optimization	+12%	+35.5 %	High
Graph models of metric interdependencies	+15%	+19.1 %	High
Meta-learning for forecast adaptation	+14.5%	+36.8 %	High

The probabilistic ensemble with a weighting mechanism demonstrated a 12% increase in accuracy and a 25% reduction in time, making it an effective compromise between speed and quality of predictions. Adaptive correction through Bayesian noise protection had the smallest effect on speed (+2.6%), but improved the accuracy of predictions by 15%, which indicates its usefulness in situations where the priority is to reduce the error rather than speed up calculations. Without the use of methods, accuracy remained basic, forecasting time did not decrease, and interpretability remained low, which confirms the need to use the proposed approaches. The most effective methods were meta-learning and hybrid optimization, which demonstrated a balance between accuracy, speed, and interpretability of predictions. Graph models also proved to be effective, especially for increasing the interpretability of results.

The most effective methods were meta-learning and hybrid optimization, which demonstrated a balance between accuracy, speed, and interpretability of predictions. Graph models also proved to be effective, especially for increasing the interpretability of results. The probabilistic ensemble and adaptive Bayesian correction further confirmed their role in reducing errors and stabilizing forecasts.

The comparative analysis of existing forecasting methods confirmed their limitations in terms of adaptability and efficiency. The developed integrated forecasting approach, which combines Bayesian noise correction, probabilistic ensembles, hybrid optimization, graph modeling, and meta-learning, was successfully validated on the GitLab CE system. Experimental evaluation demonstrated significant improvements in accuracy, execution speed, and reliability, thus meeting the requirement to enhance forecasting robustness under real operational conditions. These findings confirm that the proposed approach directly supports the main research goal – improving the reliability, efficiency, and adaptability of quantitative quality forecasting for complex software systems.

5. Conclusions

This study proposes an integrated approach for the synthesis and comparative evaluation of methods for assessing and forecasting quantitative software quality characteristics, ensuring consistency with the article's title and research objectives. The proposed framework unifies adaptive Bayesian noise correction, probabilistic ensembles with weighted risk adjustment, hybrid multi-criteria optimization, graph models of inter-metric dependencies, and meta-learning for forecast adaptation. The scientific novelty lies in the ensemble integration and adaptive mechanisms that jointly enhance robustness, interpretability, and adaptability in dynamic IT environments. Thus, the experimental validation confirms the theoretical framework (formulas 1–34), showing that Bayesian correction (24–25), ensemble synthesis (26–28), hybrid optimization (29), graph models (30–32), and meta-learning (33–34) are not only mathematically grounded but also practically effective.

The results of the GitLab CE case study confirmed the effectiveness of the integrated approach through measurable improvements: forecast error was reduced from 18.7 % to 4.2 %, execution time decreased by up to 36.8 %, and system performance metrics (CPU/RAM utilization, downtime, fault tolerance) improved between 20 % and 75 %. These findings demonstrate that the proposed approach significantly strengthens the reliability, efficiency, and adaptability of software quality forecasting compared to traditional methods.

In addition, the approach is consistent with recent research trends in software risk prediction using machine

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

learning [10] and AI-based code quality analysis with transformer models [11], which confirms the relevance of integrating advanced statistical, AI, and optimization techniques into a unified forecasting methodology. Furthermore, the proposed approach aligns with modern quality assessment standards such as ISO/IEC 25010 (quality models), ISO/IEC 25023 (measurement), ISO/IEC 5055 (structural quality measures), IEEE 730 (software quality assurance), and NIST SP 800-55 (metrics governance), ensuring methodological rigor and international applicability.

Future research should focus on extending adaptive algorithms, refining graph-based dependency modeling, and enhancing model interpretability to support wider applicability in cloud, IoT, and safety-critical domains.

Contribution of authors: goal and tasks formulation, analysis of publications, analysis of methods for predicting quantitative characteristics of software systems quality and development of an integrated approach to the synthesis of forecasting methods, writing a draft, preparing a list of references – **Anton Shantyr**; analysis of publications, formulation of tasks, analysis of modern technologies for predicting qualitative characteristics, development of graph models of relationships between metrics and meta-learning for adapting forecasts, writing drafts, preparation of a list of references, translation, general review and editing, validation – **Olha Zinchenko**; formulation of the research concept, analysis of publications, translation, formulation of conclusions, preparation of a list of references, general review and editing, validation – **Kamila Storchak**; conducting practical research on the accuracy of forecasting and forecast execution time using the proposed methods in the GitLab CE system, validation – **Viktor Vyshnivskyi**; literature search in databases, review and editing – **Olga Morozova**.

Conflict of Interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, author ship or otherwise, that could affect the research and its results presented in this paper.

Financing

This study was conducted without financial support.

Data Availability

The manuscript contains no associated data.

Use of Artificial Intelligence

The authors confirm that they did not use artificial intelligence methods while creating the presented work.

Acknowledgments

We would like to express our sincere gratitude to our colleagues from the State University of Information and Communication Technologies for their valuable support and assistance throughout this research.

All the authors have read and agreed to the published version of this manuscript.

References

1. Almetwaly, A. A. I., & Fadhel, I. E. I. Integrate between information systems engineering and software engineering theories for successful quality engineering measurement of software: Valid instrument pre-results. *Computer Software and Media Applications*, 2024, vol. 6, iss. 1, article no. 3382. DOI: 10.24294/csma.v6i1.3382.

2. Côté, P.-O., Nikanjam, A., Bouchoucha, R., Basta, I., Abidi, M., & Khomh, F. Quality issues in machine learning software systems. *Empirical Software Engineering*, 2024, vol. 29, article no. 149. DOI: 10.1007/s10664-024-10536-7.

3. Pfeiffer, R.-H., & Aaen, J. Tools for monitoring software quality in information systems development and maintenance: Five key challenges and a design proposal. *International Journal of Information Systems and Project Management*, 2024, vol. 12, iss. 1, pp. 19-40. DOI: 10.12821/ijispm120102.

4. De Angelis, G., Do, H., & Nguyen, B. N. Introduction to the special issue: "software quality for modern systems". *Journal of Software: Evolution and Process*, 2024, vol. 36, iss. 8, article no. e2663. DOI: 10.1002/smr.2663.

5. Ali, A. R., & Saleem, N. N. Classification of software systems attributes based on quality factors using linguistic knowledge and machine learning: A review. *Journal of Education and Science*, 2022, vol. 31, iss. 3, pp. 66–90. Available at: https://www.academia.edu/100852276/Classification_of_Software_Systems_attributes_based_on_quality_factors_using_linguistic_knowledge_and_machine_learning_A_review (accessed 10 Jan. 2025).

6. Ali, M. A., Yap, N. K., Ghani, A. A. A., Zulzalil, H., Admodisastro, N. I., & Najafabadi, A. A. A systematic mapping of quality models for AI systems, software and components. *Applied Sciences*, 2022, vol. 12, iss. 17, article no. 8700. DOI:10.3390/app12178700.

7. Danyk, Y., & Vysochanska, V. Technique of testing cyber vulnerabilities and quality of Cyber-physical software systems. *Theoretical and Applied Cybersecurity*, 2022, vol. 3, iss. 1. DOI: 10.20535/tacs.2664-29132021.1.251314.

8. Koi-Akrofi, G. Y., Tanye, H., Quist, S. C., Koi-Akrofi, J., Gaisie, E., Agangiba, M., & Yeboah, D. Towards a software quality factor assessment model for learning management systems (LMS). *Indian Journal of Science and Technology*, 2024, vol. 17, iss. 23, pp. 2463–2468. DOI: 10.17485/ijst/v17i23.1006.

9. Almogahed, A., Omar, M., Zakaria, N. H., Muhammad, G., & AlQahtani, S. A. Revisiting scenarios of using refactoring techniques to improve software systems quality. *IEEE Access*, 2023, vol. 11, pp. 28800-28819. DOI: 10.1109/access.2022.3218007.

10. Mahmud, M. H., Nayan, M. T. H., Ashir, D. M. N. A., & Kabir, M. A. Software Risk Prediction: Systematic Literature Review on Machine Learning Techniques. *Applied Sciences*, 2022, vol. 12, iss. 22, article no. 11694. DOI: 10.3390/app122211694.

11. Ali, I., Rizvi, S. S. H., & Adil, S. H. Enhancing Software Quality with AI: A Transformer-Based Approach for Code Smell Detection. *Applied Sciences*, 2025, vol. 15, iss. 8, article no. 4559. DOI: 10.3390/app15084559.

12. Nguyen, P. H., Sen, S., Jourdan, N., Cassoli, B., Myrseth, P., Armendia, M., & Myklebust, O. Software engineering and AI for data quality in cyber-physical systems - SEA4DQ'21 workshop report. *ACM SIGSOFT Software Engineering Notes*, 2022, vol. 47, iss. 1, pp. 26–29. DOI: 10.1145/3502771.3502781.

13. Oh, H.-S., Kim, D., & Lee, S. Review on the quality attributes of an integrated simulation software for weapon systems. *Journal of the Korean Military Science and Technology Society*, 2021, vol. 24, iss. 4, pp. 408–417. DOI: 10.9766/kimst.2021.24.4.408.

14. Jin, W., & Wang, Z. Research and application of quantitative evaluation method based on product quality characteristics. *BCP Business & Management*, 2022, vol. 17, pp. 84-90. DOI: 10.54691/bcpbm.v17i.374.

15. Pecorelli, F., Palomba, F., & De Lucia, A. The relation of test-related factors to software quality: A case study on apache systems. *Empirical Software Engineering*, 2021, vol. 26, article no. 18. DOI: 10.1007/s10664-020-09891-y.

16. Salomón, S., Duque, R., Bringas, S., & de Oliveira, K. M. Quality-in-Use in practice: A study for context-aware software systems in pervasive environments. *Journal of Software: Evolution and Process*, 2025, vol. 37, iss. 1, article no. e2764. DOI: 10.1002/smr.2764.

17. Setiadi, D., Sumitra, T., Karim, A., & Ritzkal. Software quality measurement analysis on academic information systems. *Ingénierie des systèmes d'information*, 2024, vol. 29, iss. 4, pp. 1453–1460. DOI: 10.18280/isi.290418.

Received 17.03.2025, Accepted 25.08.2025

ІНТЕГРОВАНИЙ ПІДХІД ДО ПРОГНОЗУВАННЯ ЯКОСТІ ПРОГРАМНИХ СИСТЕМ ІЗ ВИКОРИСТАННЯМ БАЙЄСІВСЬКОЇ КОРЕКЦІЇ, БАГАТОКРИТЕРІАЛЬНОЇ ОПТИМІЗАЦІЇ ТА МЕТА-НАВЧАННЯ

А. С. Шантир, О. В. Зінченко, К. П. Сторчак, В. В. Вишнівський, О. І. Морозова

Метою дослідження є вдосконалення сучасних методів прогнозування кількісних характеристик якості програмних систем для підвищення їхньої надійності, ефективності та адаптивності в динамічних IT-середовищах. Для цього було розроблено інтегрований підхід, який поєднує адаптивну байєсівську корекцію шумів, ймовірнісні ансамблі з ваговим ризиковим коригуванням, гібридну багатокритеріальну оптимізацію, графові моделі взаємозалежностей метрик та мета-навчання для адаптації прогнозів. Наукова новизна роботи полягає у запропонованій інтеграції ансамблевих і адаптивних механізмів, що підвищують стійкість прогнозування з урахуванням невизначеностей та складних залежностей між метриками. Ефективність методів була перевірена на прикладі системи GitLab CE. Експериментальні результати підтвердили суттєве покращення: похибка прогнозу знижена з 18,7% до 4,2%, час виконання прогнозу скорочено на 36,8%, споживання обчислювальних ресурсів (ЦП і пам’ять) зменшено до 20%, а показники надійності системи (відмовостійкість, тривалість простою) покращено більш ніж на 60%. Отримані метрики підтверджують, що запропонований підхід підвищує надійність, ефективність та адаптивність процесу прогнозування якості програмних систем порівняно з традиційними методами.

Ключові слова: прогнозування якості програмного забезпечення; ймовірнісні ансамблі; Байєсівська корекція; мета-навчання; багатокритеріальна оптимізація.

Шантир Антон Сергійович – канд. техн. наук, доц. каф. штучного інтелекту, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Зінченко Ольга Валеріївна – д-р техн. наук, доц., зав. каф. штучного інтелекту, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

Сторчак Каміла Павлівна – д-р техн. наук, проф., зав. каф. інформаційних систем та технологій, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

Вишнівський Віктор Вікторович – д-р техн. наук, проф., зав. каф. комп'ютерних наук, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

Морозова Ольга Ігорівна – д-р техн. наук, проф., проф. каф. комп'ютерних систем, мереж і кібербезпеки, Національний аерокосмічний університет «Харківський авіаційний інститут», Харків, Україна.

Anton Shantyr – PhD, Associate Professor at the Department of Artificial Intelligence, State University of Information and Communication Technologies, Kyiv, Ukraine,
e-mail: a.shantyr@duikt.edu.ua, ORCID: 0000-0002-0466-3659, Scopus Author ID: 57193522474.

Olha Zinchenko – Doctor of Technical Science, Associate Professor, Head of the Department of Artificial Intelligence, State University of Information and Communication Technologies, Kyiv, Ukraine,
e-mail: o.zinchenko@duikt.edu.ua, ORCID: 0000-0002-3973-7814, Scopus Author ID: 57208631196.

Kamila Storchak – Doctor of Technical Science, Professor, Head of the Department of Information Systems and Technologies, State University of Information and Communication Technologies, Kyiv, Ukraine,
e-mail: k.storchak@duikt.edu.ua, ORCID: 0000-0001-9295-4685, Scopus Author ID: 35868188400.

Viktor Vyshnivskyi – Doctor of Technical Science, Professor, Head of the Department of Computer Science, State University of Information and Communication Technologies, Kyiv, Ukraine,
e-mail: v.vyshnivskyi@duikt.edu.ua, ORCID: 0000-0003-1923-4344, Scopus Author ID: 57203266737.

Olga Morozova – Doctor of Technical Science, Professor, Professor at the Department of Computer Systems, Networks and Cybersecurity, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine,
e-mail: o.morozova@csn.khai.edu, ORCID: 0000-0001-7706-3155, Scopus Author ID: 57194517520.