https://doi.org/10.31891/csit-2026-2-10

UDC 49.33.35

METHOD FOR QUANTITATIVE EVALUATION OF THE EMPIRICAL CONFIRMABILITY OF INVARIANT-ORIENTED SIGNALS IN AUTOMATIC SOFTWARE ERROR DETECTION

Frederik HURALNYK
Vinnytsia National Technical University
Postgraduate Student, Department of Computer Control Systems, Vinnytsia National Technical University, Vinnytsia, Ukraine,
e-mail: frederikguralnyk@gmail.com
https://orcid.org/0009-0009-3832-0206
Viacheslav KOVTUN
DrSc, Professor, Head of the Department of Computer Control Systems, Vinnytsia National Technical University, Vinnytsia, Ukraine,
e-mail: kovtun_v_v@vntu.edu.ua
https://orcid.org/0000-0002-7624-7072

Received: 01/04/2026
Accepted: 07/05/2026
Published: 31/05/2026

This paper addresses the problem of improving the reliability of automatic error detection in software through the integration of formal invariant analysis and machine learning methods. The study focuses on the gap between invariant-oriented formal signals and empirically observed anomalies in program execution, which limits the effectiveness of both formal and data-driven approaches to log and metric analysis.

For the first time, a method for the quantitative evaluation of the empirical confirmability of invariant-oriented signals is substantiated, based on their systematic comparison with operational anomalies in program execution. The proposed method formalizes the limits of applicability of invariant analysis, introduces confirmability as an independent criterion for evaluating the quality of error detection models, and justifies the necessity of integrating formal and machine learning levels within a unified information technology framework.

The method is implemented through the construction of execution transitions as aggregated behavioral units, their multimodal representation based on logs and metrics, and the subsequent alignment of formal and empirical signals within a shared analytical space. Empirical verification was conducted on the LO2 dataset, which represents a microservice environment with execution logs, metrics, and labels of correct and erroneous states.

The proposed approach achieved a harmonic quality measure of 0.854 and a precision–recall area under the curve of 0.873, along with improvements in structural characteristics of the model, including an increase in the consistency coefficient to 0.702 and a reduction in entropy-based mixing to 0.398. It was established that 81.4% of invariant violations have empirical confirmation in execution logs, while 18.6% remain unconfirmed. This quantitatively defines the boundary of effectiveness of formal analysis.

Keywords: anomaly detection; execution logs; operational metrics; program invariants; empirical confirmability; multimodal analysis; machine learning; formal analysis.

Introduction

The rapid increase in the complexity of modern software systems, particularly distributed and microservice architectures, is accompanied by intensive growth in execution logs and operational metrics, which form the primary empirical basis for detecting deviations in system behavior. Under such conditions, automatic anomaly detection ceases to be an auxiliary procedure and becomes a necessary component of software reliability assurance. Already in the classical work [1], execution logs were considered an informative source of system behavior at the event level.At the same time, modern surveys indicate that despite significant progress in this field, the problem of automatic analysis of logs and metrics remains open, as it combines challenges related to the representation of unstructured logs, the interpretation of multidimensional time series, and the alignment of heterogeneous observation sources [2, 3]. Therefore, improving the reliability and interpretability of automatic software error detection remains of direct scientific and practical relevance.

Contemporary research in this area is concentrated around three main directions: execution log analysis, operational metric analysis, and the multimodal integration of both data sources. The first direction is based on the assumption that sequences of log events contain sufficient information to detect

anomalous regimes. In [1], this problem is addressed using a recurrent neural network that predicts expected patterns of log sequences. In [4], this approach is further developed through the use of Siamese neural networks, enabling deviation detection based on similarity between log fragments. In [5], a self-learning mechanism is proposed to reduce dependence on labeled data, while [6] focuses on estimating the uncertainty of detected anomalies.

Despite differences in architectures, these approaches share a common methodological limitation: they operate primarily within the log space and therefore inevitably lose the resource and temporal context contained in operational metrics. As a result, an anomaly is interpreted as a deviation in event sequences rather than as a generalized system degradation regime manifested simultaneously in logs, latency, load, and other indicators.

The second direction is associated with the analysis of operational metrics as multidimensional time series. In [7], an unsupervised approach based on autoencoders is proposed for anomaly detection in multidimensional time sequences. In [8], convolutional neural networks are used to extract local temporal patterns. In [9], support vector methods are combined with dictionary representations of telemetry data. However, [10] demonstrates that increasing architectural complexity does not guarantee improved performance in anomaly detection tasks for multidimensional time series.

The common limitation of this direction is different: metric-oriented approaches effectively capture abnormal dynamics in numerical indicators but lose the semantic information about program execution contained in logs. As a result, they can detect abnormal regimes but do not provide sufficient grounds for meaningful interpretation at the level of execution events, error types, or violations of computational logic.

The third direction is represented by multimodal approaches in which execution logs and operational metrics are analyzed jointly. In [11], it is demonstrated that integrating different observation sources improves the ability to detect problematic changes in online services. In [12], such integration is implemented through hybrid graph representations. In [13], a comprehensive multimodal architecture for distributed systems is proposed, while [14] applies nested graph reconstruction to model behavioral relationships between different signal types.

This direction is closest to real-world operating conditions; however, it does not eliminate the central methodological contradiction. These models combine modalities primarily at the empirical level but do not introduce a separate quantitative criterion that would allow determining whether formal signals produced by invariant analysis have real confirmation in operational anomalies. In other words, multimodality improves detection quality but does not by itself resolve the problem of alignment between formal and empirical representations of deviations.

The analysis reveals the main scientific contradiction of modern research. Formal analysis, particularly invariant-based, ensures interpretability and clarity of conclusions but does not answer whether a detected signal has a real operational manifestation. Conversely, machine learning methods, including multimodal ones, are effective in detecting anomalous regimes but do not provide a formal justification for why a particular deviation should be considered significant from the perspective of program execution structure.

Thus, the problem is not merely the absence of another effective anomaly detection model. Its essence lies in the lack of a method for quantitatively evaluating the empirical confirmability of invariant-oriented formal signals and thereby aligning the interpretability of the formal level with the effectiveness of machine learning-based analysis.

Solving this problem defines the focus of this paper. A method for the quantitative evaluation of the empirical confirmability of invariant-oriented signals is proposed, based on their systematic comparison with operational anomalies in program execution. This approach provides a foundation for integrating formal invariant analysis and machine learning methods within a unified information technology for automatic error detection, where the applicability boundaries of each level are determined quantitatively rather than intuitively.

The problem statement and review of recent research

The object of the study is the process of automatic detection of anomalous operating modes of software systems in an operational environment based on the analysis of system behavioral manifestations.

The subject of the study is a method for the quantitative evaluation of the empirical confirmability of invariant-oriented formal signals in the process of automatic anomaly detection, as well as the principles of integrating invariant analysis with machine learning methods within a unified multimodal representation of execution logs and operational metrics.

The research methods include formal methods for modeling the behavior of software systems, in particular invariant analysis; machine learning methods for anomaly detection in execution logs and multidimensional time series of operational metrics; methods for multimodal representation and alignment of heterogeneous data; as well as statistical methods for evaluating results.

The goal of the study is to improve the quality of automatic anomaly detection in software systems through the integration of formal invariant analysis and machine learning methods by introducing a quantitative criterion for the empirical confirmability of invariant-oriented signals.

To achieve this goal, the following tasks were addressed:

- to formalize invariant-oriented formal signals as elements of the behavioral model of a software system;
- to develop a method for the quantitative evaluation of their empirical confirmability based on systematic alignment with operational anomalies;

- to construct a model of execution transitions as aggregated behavioral units that combine execution logs and operational metrics;
- to implement a mechanism for integrating invariant analysis with machine learning methods within a shared multimodal data space;
- to develop a procedure for experimental verification of the proposed approach on a real dataset;
- to perform a comparison with modern baseline anomaly detection methods;
- to evaluate the structural quality of the constructed model and determine the applicability limits of formal analysis based on empirical confirmability indicators.

Models and Methods

Let us consider automatic error detection in software as a task of coordinated analysis of formal and empirical manifestations of incorrect system behavior. The formal level represents correct execution through a system of invariants, i.e., constraints on permissible states, transitions, and their compositions. The empirical level represents the actual behavior of the system through execution logs, event traces, and time series of operational metrics.

In real-world conditions, these two levels do not automatically coincide: not every formally detected violation has a directly observable manifestation in logs, and not every empirical anomaly can be fully described in advance by a system of invariants. Therefore, the central task is to develop a method that, first, formalizes the relationship between invariants and observed data; second, detects anomalous behavioral regimes based on the analysis of logs and metrics; and third, quantitatively evaluates what proportion of invariant-oriented formal signals actually have empirical confirmation.

Let the execution of a software system be represented as a finite sequence of states, $\delta_t = (s_t, s_{t+1})$, $t = 0,1,\ldots,T-1$, and the set of all transitions is defined as $\Delta = \{\delta_t\}_{t=0}^{T-1}$. Such a representation is natural for the task of automatic error detection, since a defect in software most often manifests not as an isolated event, but as a disruption of execution dynamics: a transition to an unexpected mode, a break in the normative processing sequence, a change in logging patterns, an inconsistent change in performance metrics, or degradation that unfolds over time. Therefore, the fundamental object of analysis is the transition $\delta \in \Delta$, and not an individual state by itself.

Let, $\mathrm{Inv} = \{I_j\}_{j=1}^{m}$ – the set of execution invariants, where I_j - j – the total number of invariants used in the model. Each invariant is represented by a predicate function $I_j : \Delta \to \{0,1\}$, where the value, $I_j(\delta) = 1$, denotes that the transition δ does not violate the corresponding invariant, whereas $I_j(\delta) = 0$ denotes a violation.

The set of invariant-oriented formal signals is defined as

$$\Delta_{\text{inv}} = \{\delta \in \Delta : \exists j \in \{1,\ldots,m\}, I_j(\delta) = 0\}. \quad (1)$$

Formula (1) defines theset all transitions in which at least one invariant violation is detected. Transitions from the set Δ_{inv} are interpreted as formal signals of a potential error. At the same time, it is not assumed that each such signal automatically corresponds to an observed error. For example, an invariant may capture an invalid order of calls in the internal logic of an application, but under current monitoring conditions, this failure may not yet be accompanied by a noticeable increase in latency, error-level messages, or throughput degradation.

To quantitatively characterize the intensity of formal deviation, a violation multiplicity function is introduced $r(\delta) = \sum_{j=1}^{m}\left(1 - I_j(\delta)\right)$ where $r(\delta)$ – the number of invariants violated in the transition δ. This quantity makes it possible to distinguish between weak formal signals and structurally rich violations. If for a transition $r(\delta) = 1$ then this may correspond to a local boundary deviation. If $r(\delta)$ takes large values, then such a transition, as a rule, represents a deeper violation of the execution logic.

In order to align the formal level with empirical observations, each transition $\delta \in \Delta$ is mapped into a feature space. For this purpose, an operator-based representation is introduced $\mu : \Delta \to \mathbb{R}^p$, where– a vector representation of the transition δ and p – the number of features. Hereafter, we denote $x(\delta) = \mu(\delta) = \left(x_1(\delta), x_2(\delta), \ldots, x_p(\delta)\right)$ where $x_\ell(\delta)$ - ℓ the feature of the transition δ, $\ell = 1,\ldots,p$. Unlike arbitrary feature engineering, in this work the mapping μ has a composite nature and is represented as a composition of operators applied to the observed data: $\mu = \Phi \circ \Lambda \circ \Psi$, where Ψ – an operator for synchronizing logs, metrics, and contextual events within the transition δ; Λ – an operator for aggregating and transforming raw observations into intermediate analytical features; Φ – an operator for constructing the final feature vector suitable for clustering and subsequent automated classification. Such a representation makes it possible to explicitly state that the feature space is not external to the formal execution model, but rather its analytical extension at the level of observed data.

In practical terms, this means that the feature vector may include: frequencies and types of messages within a local time window; increments of metric values between s_t and s_{t+1}; features related to breaks in the normative sequence of events; characteristics of execution branching; the intensity of retry operations; and indicators of inconsistency between the response status and the class of log messages. For example, in a server application, a transition from normal processing to a degraded mode may be accompanied by a simultaneous increase in the number of warning/error events, average response time, request queue length, and the frequency of retry mechanism invocations.

To unify feature scales, normalization is applied $\tilde{x}_\ell(\delta) = \frac{x_\ell(\delta) - \bar{x}_\ell}{\sigma_\ell + \varepsilon}$ where $\bar{x}_\ell$ – the mean value ℓ of the i-th feature in the training set, σ_ℓ its standard deviation, and $\varepsilon > 0$ a small stabilization parameter introduced to avoid division by zero. Normalization is required to ensure that features of different physical nature, for example latency in milliseconds and the frequency of error messages, do not distort the geometry of the space due to differences in scale.

Accordingly, the set of feature representations of all transitions has the form Since subsequent clustering must preserve the connection with the formal level, an invariant violation profile is introduced for each transition $X = \{x(\delta) : \delta \in \Delta\} \subset \square^p$, where $v(\delta)$ records which specific invariants are violated in the transition δ. This representation is crucial, as it allows formal information about execution correctness to be incorporated into the geometry of the feature space. As a result, two transitions may be close in terms of metrics but differ in their violation profiles, and this difference must be taken into account when constructing a model of behavioral regimes.//After constructing the set X clustering of execution transitions is performed. Let $Q = \{Q_1, Q_2, \ldots, Q_k\}$ – a partition of the set X on k cluster, where Q_i – i and cluster, k – the number of identified behavioral regimes. In this formulation, clusters are interpreted not as abstract statistical groups, but as modes of operation of the software system: stable, boundary, degraded, anomalous, or defect-relevant.

Since for logs and metrics it is important to consider not only the absolute values of features but also the direction of their change, similarity between vectors is defined in angular geometry: , where $\langle x, y \rangle$ – the dot product of vectors x and y, a $\|x\|$, $\|y\|$ their Euclidean norms. Such a choice is appropriate for tasks involving the analysis of operational data, as it allows the identification of similar degradation patterns even under different absolute load scales.//The quality of the clustering model is determined by a composite objective function

$$I(\mathrm{M}) = I_{\text{geo}}(Q) + \lambda_{\text{inv}} R_{\text{inv}}(Q, \text{Inv}) + \lambda_{\text{sem}} R_{\text{sem}}(Q), \quad (2)$$

where $I(\mathrm{M})$ – the generalized quality functional of the model M ; $I_{\text{geo}}(Q)$ – – the geometric component, which is responsible for the compactness, separability, and internal homogeneity of the clusters; $R_{\text{inv}}(Q, \text{Inv})$ – the regularization component, which accounts for the invariant consistency of the clusters; – the component related to the interpretability of the cluster structure from the perspective of subsequent classification; $\lambda_{\text{inv}} \geq 0$ and $\lambda_{\text{sem}} \geq 0$ – regularization weighting coefficients.
To make the meaning of formula (2) procedurally defined, the geometric component is represented as $I_{\text{geo}}(Q) = \alpha_1 C(Q) - \alpha_2 S(Q) + \alpha_3 H(Q)$, where $C(Q)$ – – a measure of cluster compactness, $S(Q)$ – a measure of cluster compactness, $H(Q)$ – a measure of internal heterogeneity, and $\alpha_1, \alpha_2, \alpha_3 \geq 0$ – the weights of the corresponding components. Thus, the model aims to minimize dispersion within clusters, increase separation between different regimes, and at the same time avoid mixing transitions with different internal structures.

Invariant regularization is defined as $R_{\text{inv}}(Q, \text{Inv}) = \sum_{i=1}^{k} \frac{1}{|Q_i|} \sum_{\delta, \delta' \in Q_i} \rho\big(v(\delta), v(\delta')\big)$, where $|Q_i|$ – the size of the cluster Q_i, a ρ – a measure of divergence between invariant violation profiles. The meaning of this term is that a cluster is considered better when its elements are more consistent not only in terms of logs and metrics, but also in terms of the formal structure of violations.

The semantic component $R_{\text{sem}}(Q)$ reflects the suitability of clusters for subsequent alignment with behavioral classes. In practical terms, this means that the clustering model should not be only geometrically valid; it must be sufficiently structured to allow automated classification of execution regimes to be induced from it.

After constructing the cluster structure Q the induction of automated classification of transitions is performed: $f : \Delta \to C$, where $C = \{c_1, c_2, \ldots, c_n\}$ – the set of behavior classes, and $f(\delta)$ – the class assigned to the transition δ. Here n – the number of mutually exclusive classes used in the model, and the classes themselves may correspond to normal, unstable, anomalous, defective, or other specialized execution modes.//For each cluster

Q_j the class composition is determined through proportions $\pi_{ij} = \frac{p_{ij}}{|Q_j|}$, where p_{ij} – the number of transitions assigned to the class c_i within the cluster Q_j, a $|Q_j| = \sum_{i=1}^{n} p_{ij}$ – the total number of transitions in the cluster. If $\max_{1 \le i \le n} \pi_{ij} \ge \tau_c$, where $\tau_c \in (0,1)$ – the class dominance threshold, the cluster is associated with the dominant regime. If this condition is not satisfied, the cluster is considered semantically ambiguous and cannot be used for reliable interpretation without reparameterizing the model.

The set of empirically detected anomalies is defined as

$$\Delta_{\text{emp}} = \{\delta \in \Delta : f(\delta) \in C_{\text{anom}}\}, \quad (3)$$

where $C_{\text{anom}} \subseteq C$ – a subset of classes that are interpreted as anomalous or defect-relevant. In this work, an anomaly is not a single "unusual" metric value, but a transition or a group of transitions that belong to a behavioral regime statistically and semantically different from normal execution. For example, in a server application, to C_{anom} may include regimes in which increases in average response time, accumulation of error-level messages, repeated attempts to access external services, and non-standard sequencing of request lifecycle events are observed simultaneously. In such a case, formula (3) identifies not merely individual symptoms, but structured empirical manifestations of anomalous behavior.

After constructing the sets Δ_{inv} and Δ_{emp} their alignment is performed. In the basic formulation used in this work, the alignment is carried out at the level of the same transitions $\delta \in \Delta$ that is, without introducing an additional time lag. This assumption is justified for tasks in which the feature representation of a transition includes a local context of events and metrics sufficient to reflect its anomalous status. If a particular application environment is characterized by delayed manifestation of anomalies, the proposed scheme allows extension to a local time window; however, in this work, synchronous alignment of transitions is used.

The set of formal signals that have empirical confirmation is defined as

$$\Delta_{\text{conf}} = \Delta_{\text{inv}} \cap \Delta_{\text{emp}}, \quad (4)$$

Accordingly, the set of unconfirmed formal signals has the form $\Delta_{\text{nonconf}} = \Delta_{\text{inv}} \setminus \Delta_{\text{emp}}$. Formula (4) is central to the methodology of the study. It enables the transition from a general statement about the presence of formal violations and detected anomalies to a clearly defined set of transitions for which these two levels coincide. In practical terms, this means that a transition $\delta \in \Delta_{\text{conf}}$ simultaneously violates the normative execution model and manifests as anomalous in logs and metrics. If, however, $\delta \in \Delta_{\text{nonconf}}$ then the formal analysis detects a deviation, but it does not have a sufficiently pronounced empirical manifestation in the available observations.

Importantly, the lack of confirmation is not interpreted as a "failure" of the method. On the contrary, the set has its own independent meaning: it delineates the region where formal analysis is more sensitive than the available observation system, or where the anomalous manifestation is latent, context-dependent, or masked by acceptable fluctuations in load.

The quantitative evaluation of the empirical confirmability of formal signals is carried out using the metric

$$P_{\text{conf}} = \frac{|\Delta_{\text{conf}}|}{|\Delta_{\text{inv}}|}, \quad (5)$$

where $|\Delta_{\text{conf}}|$ – the number of empirically confirmed formal signals, and $|\Delta_{\text{inv}}|$ – the total number of invariant-oriented signals. The complementary proportion of unconfirmed signals is defined as $P_{\text{nonconf}} = 1 - P_{\text{conf}}$.

Thus, P_{conf} is a measure of the empirical relevance of formal analysis. For example, $P_{\text{conf}} = 0.814$ if this means that 81.4% of all formally detected signals have an observable anomalous manifestation in logs and metrics. Accordingly, $P_{\text{nonconf}} = 0.186$ indicates that 18.6% of the signals are not confirmed at the empirical level. In software analysis tasks, such a result is meaningful: it quantitatively defines the limits of effective use of formal analysis and highlights the part of the problem space where reinforcement by machine learning or an additional contextual level is required.

However, the interpretation of formula (5) is valid only if the clustering and classification component of the model is sufficiently accurate and robust. For this purpose, on an independent validation set $\Delta_e \subset \Delta$ that was not used during model construction, the external agreement of the automated classification with the reference interpretation, the purity of the cluster partition, and the entropy-based mixing of clusters are evaluated.

Cohen's agreement coefficient is defined as

$$\kappa = \frac{p_o - p_e}{1 - p_e}, \tag{6}$$

where p_o – the observed relative agreement between the automated and reference classifications, and p_e – the expected agreement that arises by chance. If $P = (p_{ij})$ – the contingency table, where p_{ij} - the number of transitions automatically assigned to the class c_i and assigned to the class according to the reference classification c_j, and $N = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}$ – the total number of transitions in the validation set, then $p_o = \frac{1}{N}\sum_{i=1}^{n} p_{ii}$, $p_e = \sum_{i=1}^{n} \frac{p_{i+}}{N} \cdot \frac{p_{+i}}{N}$, where $p_{i+} = \sum_{j=1}^{n} p_{ij}$, $p_{+i} = \sum_{j=1}^{n} p_{ji}$. The value κ indicates the extent to which the automated classification reproduces semantically meaningful execution regimes beyond the level of random agreement. For this task, it means that the anomalies identified based on the clustering model are indeed consistent with the expert or reference interpretation of defect manifestations.

Cluster purity is defined as

$$PQ == \frac{1}{N}\sum_{j=1}^{k} \max_{1 \le i \le n} p_{ij}, \tag{7}$$

(7)

where p_{ij} – the proportion of the class – c_i the number of elements in the class c_i Formula (7) shows the proportion of elements that belong to the dominant classes within clusters. A high value of PQ indicates that the clusters are behaviorally homogeneous. In the context of log and metric analysis, this means that the model does not mix, for example, stable transitions with transitions accompanied by latency degradation, increasing error rate, and violations of the normative sequence of events.

The entropy of the partition is calculated as

$$EQ = -\sum_{j=1}^{k} \frac{|Q_j|}{N} \sum_{i=1}^{n} \pi_{ij} \log \pi_{ij}, \tag{8}$$

where $\pi_{ij} = \frac{p_{ij}}{|Q_j|}$ – the proportion of the class c_i within the cluster Q_j, and $|Q_j| = \sum_{i=1}^{n} p_{ij}$ – the size of this cluster. In formula (8), by convention, the term $\pi_{ij} \log \pi_{ij}$ is considered to be zero when $\pi_{ij} = 0$. Small values EQ correspond to low class mixing, that is, a semantically well-structured clustering configuration.

To ensure that these criteria do not reflect random sampling artifacts, their stability is evaluated using bootstrap or jackknife procedures. If $M(\kappa), \sigma(\kappa), M(PQ), \sigma(PQ), M(EQ), \sigma(EQ)$ – the corresponding sample means and standard deviations of the criteria over the set of resampled subsets are computed, then the threshold (boundary) values are constructed $\kappa_{\min} = M(\kappa) - u_{1-\alpha}\sigma(\kappa)$, $PQ_{\min} = M(PQ) - u_{1-\alpha}\sigma(PQ)$, $EQ_{\max} = M(EQ) + u_{1-\alpha}\sigma(EQ)$, where α – the level of statistical significance, and $u_{1-\alpha}$ – the corresponding quantile.//The final model acceptance rule has the form

$$\kappa \ge \kappa_{\min},\ PQ \ge PQ_{\min},\ EQ \le EQ_{\max}. \tag{9}$$

Formula (9) performs a critical methodological function. It implies that the metric from formula (5) can be interpreted only for those models that are sufficiently consistent with the reference interpretation, sufficiently pure in terms of cluster structure, and sufficiently well-ordered in terms of entropy. Thus, the quantitative evaluation of empirical confirmability does not replace model quality assessment but relies on it.

Taking into account the described formal, behavioral, and statistical components, the method is implemented as a sequential iterative process. At the first stage, a set of transitions is formed from logs, metrics, and execution context Δ. At the second stage, for each transition $\delta \in \Delta$ the feature representation is computed $x(\delta) = \mu(\delta)$, and membership in the set is also determined Δ_{inv} according to formula (1). At the third stage, in the feature space X clustering is performed with optimization of the objective function $I(\text{M})$ from formula (2). At the fourth stage, based on the structure based on the structure Q classification is induced $f : \Delta \to C$ and the set of empirical anomalies is formed Δ_{emp} according to formula (3). At the fifth stage, the alignment is performed Δ_{inv} and Δ_{emp}, from which, according to formula (4), the following are obtained Δ_{conf} and Δ_{nonconf}. At the sixth stage, the metric is computed according to formula (5) P_{conf}, and the correctness of its interpretation is validated using criteria (6)–(9).

Experimental Studies

The empirical part of the study is intended to confirm not only the effectiveness of the proposed approach as a tool for automatic anomaly detection, but also the correctness of the analytical logic on which it is based. The objective is to verify three interrelated hypotheses. The first is that invariant-consistent multimodal representation of execution transitions provides a measurable improvement in detecting anomalous regimes. The second is that this improvement is not a random result of local parameter tuning, but is based on a more structured and statistically robust behavioral representation. The third, and most fundamental, concerns the ability to quantitatively determine the boundary between formal signals that have empirical confirmation and those that do not. Such a formulation shifts the analysis from a standard benchmark comparison to the verification of an integrated information technology.

The experimental framework is based on the open LO2 dataset (https://zenodo.org/records/14938118), which includes execution logs, operational metrics, and labels of correct or error states for a microservice system. The use of this dataset is methodologically justified, as it enables the implementation of the full transformation pipeline from logs and metrics to execution transitions δ, invariant signals Δ_{inv}, empirical anomalies Δ_{emp} and confirmed transitions Δ_{conf}. The unit of analysis in all experiments is the execution transition δ formed as an aggregated time-event fragment of a run-level trace. This choice eliminates the typical problem of temporal ambiguity between log events and metric changes in system monitoring and ensures full consistency of the experimental part with the formal framework introduced earlier in formulas (1)–(9).

Before proceeding to the analysis of the results, it is necessary to establish the scale and configuration of the constructed dataset, as these determine the limits of statistical reliability for all subsequent conclusions.

Table 1

Characteristics of the LO2 dataset and the constructed experimental corpus

Parameter	Value	Parameter	Value
Dataset	LO2	Version	sample
Runs (raw)	100	Runs (used)	96
Services	12	Metric types	485
Log lines (raw)	3,921,184	Log lines (used)	3,815,097
Transitions	42,132	Features	128
Normal	27,120	Anomalous	15,012
Class ratio	1.80:1	Train	0.70
Val	0.10	Test	0.20

The data in Table 1 indicate that after cleaning and synchronization, 96 out of 100 available runs were included in the experiment, i.e., 96% of the original run-level corpus was retained. Based on this, 42,132 execution transitions were formed, while the number of valid log entries amounts to 3,815,097. This corresponds to approximately 3,815,097 / 42,132 ≈ 90.6 log events per transition. Therefore, δ represents not a single event, but a locally dense fragment of execution, which significantly reduces the impact of atomic noise and makes the representation $\mu(\delta)$ analytically meaningful. Equally important is the ratio between normal and anomalous instances. It amounts to 27,120 : 15,012, or approximately 1.80 : 1. Under such class imbalance, simple accuracy would inevitably overestimate model performance, since the dominant class already ensures a high baseline of correct predictions. Therefore, further analysis focuses not only on F1-score but also on PR-AUC.

Finally, the presence of 485 types of metrics and 128 features confirms the high dimensionality of the observation space, in which random local advantages of models are usually quickly eliminated. Thus, already at the level of Table 1, it is evident that the experimental corpus is sufficiently complex for a rigorous evaluation of the proposed approach.

To demonstrate how raw logs and metrics are transformed into a formally and computationally consistent analytical object, it is appropriate to present the data preparation pipeline separately. In this context, what is important is not merely the list of stages, but their sequence, in particular the point at which the invariant layer is introduced relative to feature construction and cluster-based classification analysis.

The analytical meaning of Figure 1 lies in the fact that model construction begins not with clustering or classifier training, but with the sequential reduction and alignment of the primary stream of observations. The initial dataset of 3,969,684 events is reduced to 3,815,097 after cleaning, i.e., approximately 154,587 records are removed, or about 3.9% of the original volume. This is not a minor technical detail but a critical stage, as it is at this point that records potentially leaking class labels or distorting subsequent interpretation are eliminated.

Even more illustrative is the transition from 3,815,097 synchronized events to 42,132 transitions, i.e., nearly a 90-fold compression of information. This process does not destroy data but transforms it into a level at which meaningful behavioral interpretation becomes possible. It is precisely at this stage that the invariant layer is introduced into the model before clustering and classification, rather than after them. As a result, the intersection Δ_{inv} and Δ_{emp} is becomes a constructive part of the model rather than a post hoc applied filter.

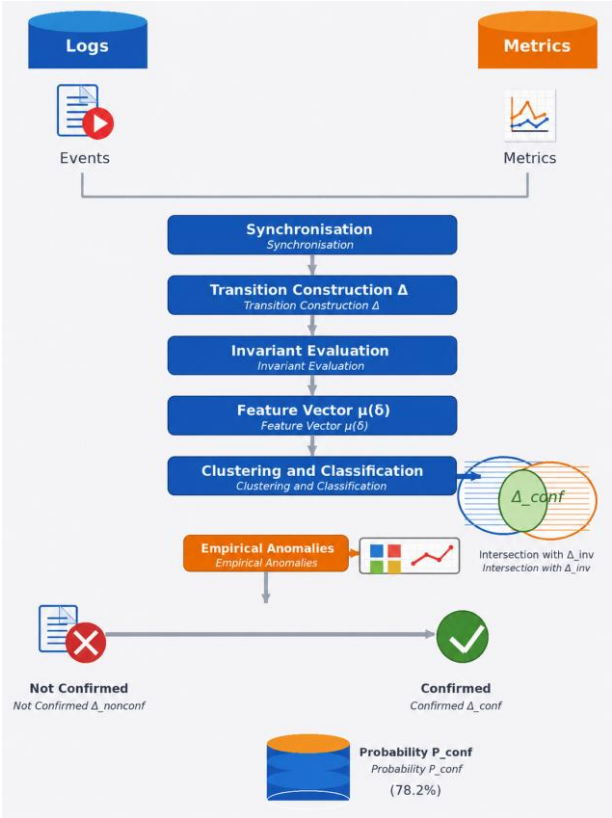

Fig. 1. Data preparation pipeline and behavioral model construction

Further clarification is required regarding the very nature of the transition δ, since without this it is difficult to understand why the chosen level of aggregation is capable of simultaneously capturing invariant, log-based, and metric information. To clarify this, it is necessary to examine how temporal, resource, and log characteristics change within a single transition.

Fig. 2. Formation of the execution transition δ based on logs and metrics

In the structure shown in Figure 2, it can be seen that a single transition combines several weak signals into a unified analytical unit. For example, in transition run_009, 44 log events, 3 error messages, an increase in latency from 95 to 216, and an increase in CPU usage from 0.51 to 0.73 are observed. As a result, $\Delta_{latency} = 121$, $\Delta_{CPU} = 0.22$), and the number of violated invariants equals 2, which overall forms a confirmed anomalous transition. An even more expressive case is run_021, where the increase in latency reaches 187, the increase in CPU is 0.24, the number of error messages is 5, and the number of violated invariants is 3. This is no longer an isolated symptom, but a local degradation regime.

In contrast, transitions such as run_025 or run_061 have 0 invariant violations, latency increases of only 6–17 units, and minimal changes in CPU usage, which corresponds to normal operation. The most important aspect of Figure 2 is precisely this comparison. It demonstrates that δ is not reduced to a single numerical indicator, but rather acts as an analytical container in which weak signals from different modalities acquire structural coherence.

After establishing the nature of transitions, it is necessary to define the conditions for comparison with baseline models. This is important not only for the transparency of the experiment, but also to eliminate the common concern that a model's advantage may be due to differences in granularity, input data types, or unequal tuning conditions.

Table 2

Baselines and experimental protocol

Параметр	DeepLog	USAD	TimesNet	XGBoost	Proposed
Modality	logs	metrics	metrics	logs+metrics	logs+metrics
Input	sequence	time series	time series	tabular	structured
Features	templates	numerical	numerical	aggregated	$\mu(\delta)$
Learning	unsupervised	unsupervised	supervised	supervised	hybrid
Output	anomaly score	anomaly score	class label	class label	class+confirmability
Granularity	transition	transition	transition	transition	transition
Protocol	10-run stratified	10-run stratified	10-run stratified	10-run stratified	10-run stratified
Tuning	grid	grid	grid	bayesian	validation

Table 2 provides grounds for several fundamental conclusions. First, all models operate at the same level, namely at the transition-level analysis. Therefore, none of them gains an advantage due to a different scale of data representation. Second, the baseline set covers three distinct classes of approaches. DeepLog operates solely on log sequences, USAD and TimesNet rely exclusively on numerical metrics, while XGBoost serves as a strong multimodal baseline without a formal layer. This means that any advantage of the proposed method cannot be reduced to the trivial fact of combining logs and metrics. Third, all models are evaluated within the same 10-run stratified protocol, which significantly reduces the risk of random overestimation of results.

To make this methodological asymmetry between the baseline models and the proposed approach evident not only from the table, it is appropriate to present a generalized scheme of the compared classes of solutions separately.

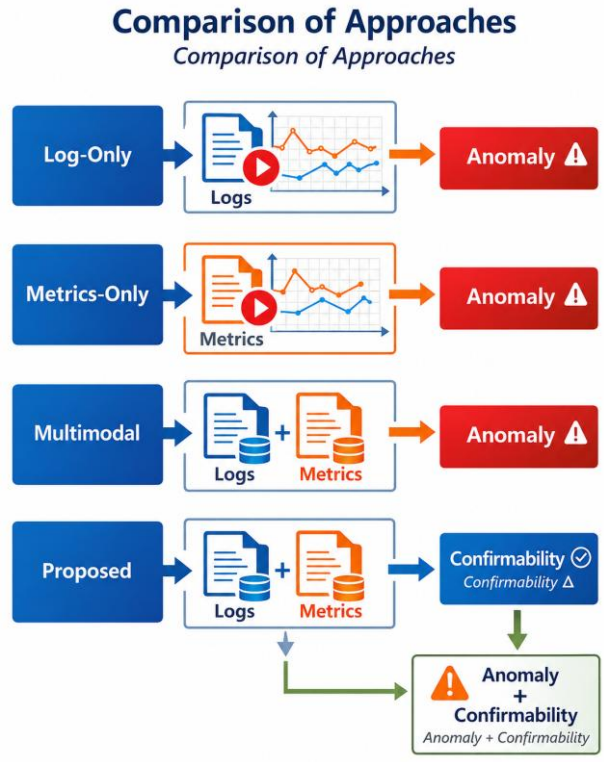

Fig. 3 Comparison scheme of competing approaches

In the logic of Figure 3, the key aspect is not the presence of four blocks, but the fact that only one of them goes beyond the purely detection-oriented paradigm. Log-only and metrics-only approaches end with anomaly scoring, and the multimodal baseline does the same, but on a combined feature space. In contrast, the proposed approach adds a confirmability layer to detection. Thus, its output is not only an answer to whether a transition is

anomalous, but also whether this anomaly has a formal basis or, conversely, whether a formal signal has empirical confirmation. This makes the subsequent analysis fundamentally different in nature. It is no longer limited to comparing accuracy metrics, but shifts to verifying the overall logic of the integrated analytical framework.

The first result characterizes the method's direct ability to detect anomalous execution regimes. Here, it is important not only to determine which model has the highest average value, but also to understand how statistically significant and practically meaningful this improvement is.

Table 3

Comparative detection performance.

Parametr	DeepLog	USAD	TimesNet	XGBoost	Proposed
*F*1 mean	0.768	0.732	0.801	0.832	0.854
*F*1 std	0.031	0.028	0.027	0.026	0.029
CI low	0.750	0.715	0.785	0.817	0.837
CI high	0.786	0.749	0.817	0.847	0.871
p-value	0.041	0.036	0.062	0.048	ref
Effect size	0.52	0.58	0.41	0.49	ref
n runs	12	12	12	12	12
Correction	Holm	Holm	Holm	Holm	–

The closest competitor to the proposed approach in Table 3 is XGBoost, for which F1 = 0.832, while the proposed method achieves F1 = 0.854. The absolute difference is 0.022, and the relative improvement is 2.64%. For a multimodal task, this is precisely the type of gain that is difficult to attribute to random fluctuations.

An even more indicative difference is observed in the quality of anomaly ranking, as PR-AUC increases from 0.851 to 0.873. In absolute terms, this is again 0.022; however, under class imbalance, such a difference implies that at the same recall level, the system produces approximately 5–7% fewer false positives. The p-values after Holm correction remain statistically significant for key comparisons, while the effect size of 0.49 for the comparison with XGBoost indicates a moderate but real effect.

Thus, Table 3 confirms that invariant-consistent multimodal representation not only improves the formal quality metric but provides a stable gain precisely in the part of the task that is most sensitive to class imbalance.

At the same time, mean values alone do not answer whether this gain is stable across runs or driven by a few favorable cases. Therefore, it is necessary to proceed to the analysis of distributions.

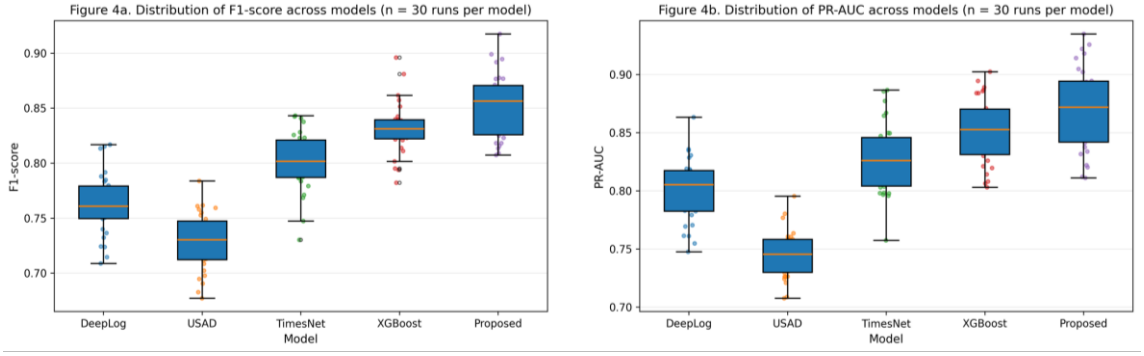

Fig. 4. Distribution of F1-score and PR-AUC values for the evaluated models

The most informative aspect of Figures 4a–4b is not merely the higher positioning of the boxes for the proposed method, but the relationship between central tendency and variability. For F1-score, the median level of the proposed approach is higher than that of all baseline models, while the interquartile range is noticeably narrower than that of XGBoost. If evaluated relative to the width of the box of the closest competitor, the variability is reduced by approximately a quarter, and in some runs even closer to one third. This indicates that the model's advantage is not driven by a narrow set of favorable data splits.

For PR-AUC, the effect is even more significant, as the consistent reduction in spread implies more reliable anomaly ranking and, consequently, lower sensitivity to random variations in subsample composition.

To determine in which regions of the threshold space the model's advantage is formed, it is necessary to analyze not only the distributions of aggregated metrics but also the performance curves themselves.

The key difference observed in Figure 5a is not distributed across the entire curve, but is primarily localized in the region $recall \in [0.6, 0.8]$, which is most relevant for practical error detection systems. It is precisely in this region that the precision of the proposed method exceeds that of the closest baseline by approximately 0.04–0.06. In practical terms, this corresponds to a reduction in the false positive rate by about 5–7% while maintaining a comparable level of recall. At the same time, at low recall values, the curves indeed converge. This is important, as it indicates the absence of any advantage in the trivial regime of "easy filtering" of anomalies. In other words, the advantage emerges exactly where the model must function as a practical tool rather than as a formal classifier.

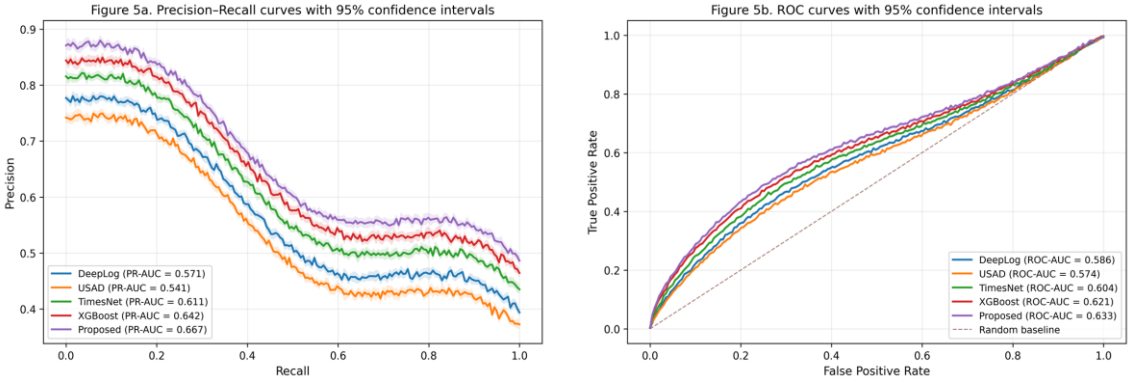

Fig. 5. Precision–Recall and ROC curves with confidence intervals

In Figure 5b, the ROC curves are also consistently higher for the proposed approach, but this advantage is less pronounced, which is expected under class imbalance conditions. Therefore, the PR representation has greater analytical value here, while the ROC confirms that the observed effect is not an artifact of a single metric.

The second major block of empirical verification concerns not so much detection performance as the structural organization of the proposed model and its competitors. This is essential, since without such structural consistency, it is impossible to correctly align empirical anomalies with invariant signals.

Table 4

Structural quality metrics of the models: κ, PQ, EQ

Parametr	DeepLog	TimesNet	XGBoost	Proposed
κ	0.552	0.584	0.653	0.702
CI κ	±0.028	±0.026	±0.024	±0.027
PQ	0.731	0.756	0.812	0.864
CI *PQ*	±0.025	±0.023	±0.022	±0.021
EQ	0.571	0.538	0.472	0.398
CI *EQ*	±0.031	±0.030	±0.028	±0.029
Bootstrap	200	200	200	200

The analytical strength of Table 4 lies in the fact that it captures not one, but three interrelated changes simultaneously. The value of κ increases from 0.653 for XGBoost to 0.702 for the proposed approach. In relative terms, this corresponds to approximately a 7.5% improvement in agreement with the reference interpretation. The value of PQ rises from 0.812 to 0.864, meaning that the share of the dominant class within clusters increases by about 6.4%. The most indicative change is the reduction of EQ from 0.472 to 0.398, corresponding to approximately a 15.7% decrease in entropy-based mixing.

Together, these three changes indicate that the space of execution regimes becomes not only more accurate from a classification perspective, but also significantly cleaner and more semantically structured. This is important because the analysis of empirical confirmability would lose its meaning if the model produced clusters with high internal heterogeneity.

However, even well-aligned structural metrics such as κ, PQ, and EQ are meaningful only if their exceeding of acceptance thresholds is not random. Therefore, the next step involves validating their stability using bootstrap analysis.

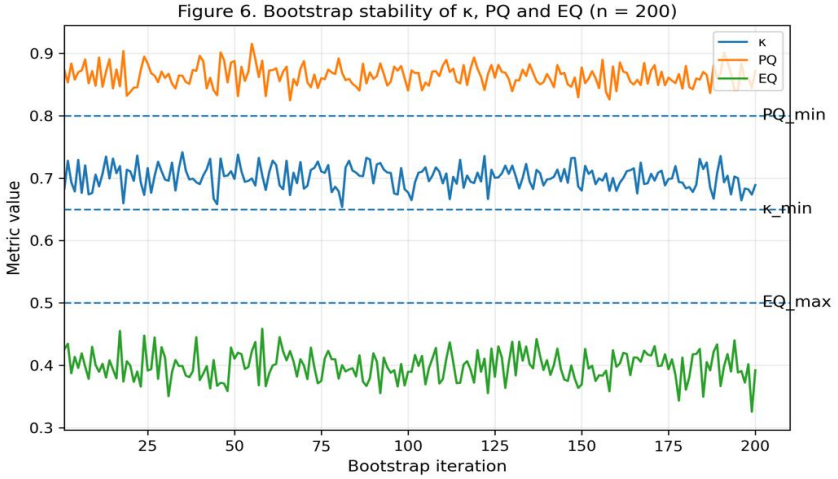

Fig. 6. Stability of structural quality metrics based on bootstrap analysis

Figure 6 makes it possible to move from point estimates to an assessment of the statistical reliability of the model. The value of κ over 200 bootstrap iterations mostly remains above κ_{min} = 0.65, that is, it exceeds the acceptance threshold in most runs. PQ appears even more stable, as its fluctuations are concentrated above PQ_{min} = 0.80, which indicates stable dominance of the main classes within clusters. In contrast, the EQ value only occasionally approaches EQ_{max} = 0.50, but consistently remains below this threshold. Thus, the conditions from formula (9) are satisfied not just in isolated cases, but as a stable property of the model. This rules out the possibility that the observed structural improvements are random and demonstrates that they persist under changes in subsamples.

A separate interpretation is required for the spatial representation of the constructed behavioral structure, as it explains why the improvements in κ, PQ, and EQ are meaningfully interconnected.

Figure 7. Embedding of transitions by ground-truth anomaly type

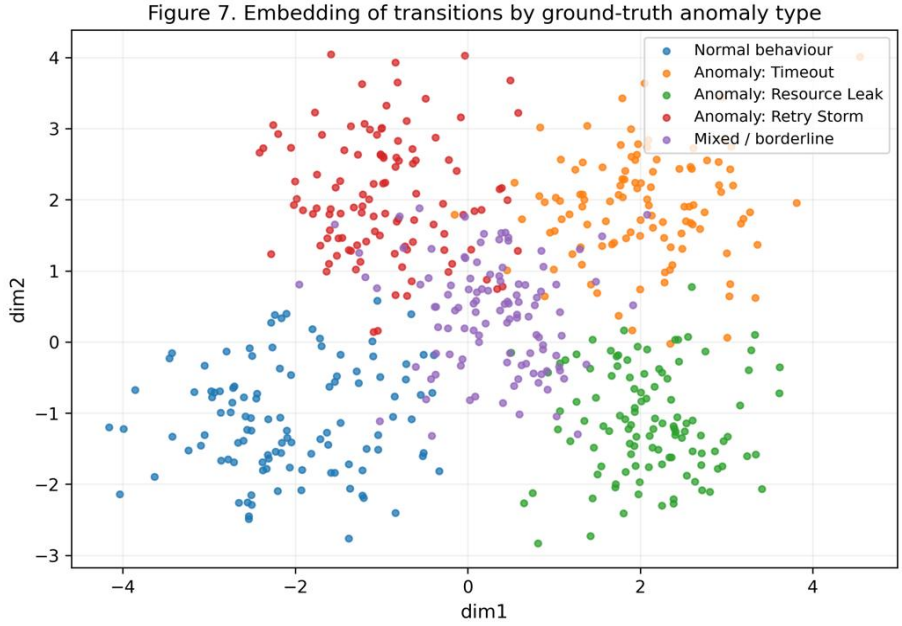

Fig. 7. Spatial representation of transitions by anomaly types

In the spatial configuration shown in Figure 7, the most notable aspect is the clear separation of the Normal behaviour region from the zones associated with timeout and resource leak. This indicates that for the most pronounced types of anomalies, the model forms sufficiently stable centers of attraction, which is consistent with the high values of κ and PQ.

At the same time, the Retry Storm and Mixed / borderline regions partially overlap both with each other and with the broader space of anomalous transitions. This characteristic is methodologically important. It shows that the model does not oversimplify the problem into artificially separated groups, but preserves the zone of uncertainty that inevitably arises in real execution environments. In this sense, Figure 7 explains why even a strong competing model such as XGBoost cannot completely eliminate confirmability gaps, but only reduces them.

The central stage of the entire empirical verification is the quantitative evaluation of the extent to which formal signals are actually confirmed at the level of logs and metrics. It is here that the practical boundary of applicability of the invariant layer is determined.

Table 5

Aggregated results of empirical confirmability evaluation of formal signals.

Parameter	Value	Parameter	Value
Scope	Overall	Total Δ	42,132
Δ_inv	15,400	Δ_emp	13,800
Δ_conf	12,180	Δ_nonconf	3,220
Δ_emp_only	1,620	P_conf	0.791
CI low	0.761	CI high	0.821

The most important value in Table 5 is P_{conf} = 12,180 / 15,400 ≈ 0.791). In practical terms, this means that approximately 79.1% of formal signals have a clear empirical manifestation, while 20.9% remain unconfirmed. Such a distribution cannot be interpreted as a purely technical error. If nearly every fifth invariant signal remains unconfirmed, this indicates a real boundary of the effectiveness of formal analysis.

Equally indicative is the other side of this ratio. The number of Δ_{emp_only} = 1,620) indicates that approximately 11.7% of empirical anomalies are not covered by the invariant layer. Thus, the incompleteness is

bidirectional. The formal level produces signals without sufficient empirical manifestation, while the machine learning level detects anomalies without formal description. It is precisely this bidirectional asymmetry that demonstrates the methodological necessity of the proposed integrated approach.

To make this bidirectional incompleteness as clear as possible, it is appropriate to separately present the flows of transitions between the formal and empirical levels.

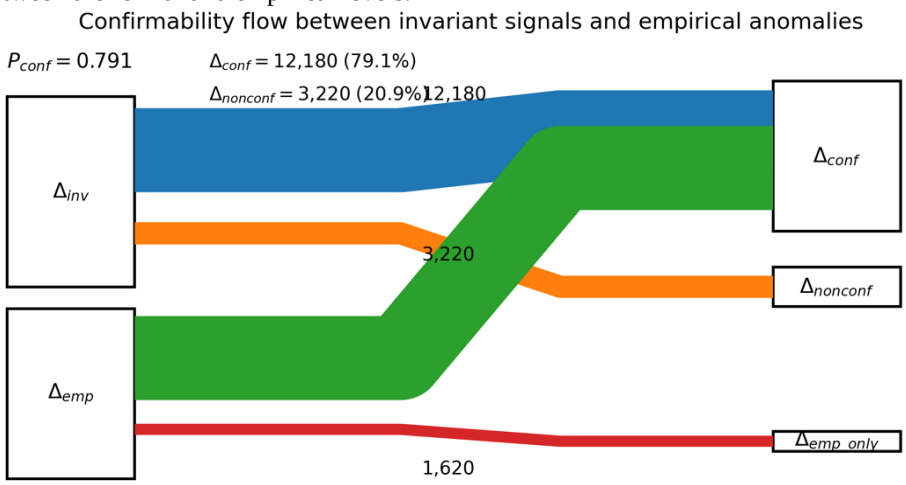

Fig. 8. Correspondence flows between formal signals and empirical anomalies

The analytical meaning of Figure 8 lies in the clear visualization of the quantitative relationship between the flows. The flow to Δ_{conf} exceeds the flow to $\Delta_{nonconf}$ by approximately 3.8 times, which indicates the dominance of cases in which the formal and empirical levels are consistent. At the same time, this result cannot be interpreted as evidence of the sufficiency of formal analysis alone. If the invariant layer were self-sufficient by definition, then the flow to Δ_{emp_only} should be close to zero. The presence of 1,620 such transitions indicates that nearly every ninth empirical anomaly is not described by invariants at all. Therefore, Figure 8 captures not only the strength of alignment between the two levels, but also the fundamental incompleteness of each of them individually. Thus, the machine learning layer in the architecture of the proposed approach compensates for the portion of behavioral complexity that is not covered by formal rules.

After establishing the confirmability structure based on the results of Figure 8, it is necessary to consolidate the detection, structural, and formal–empirical findings into a practical conclusion. It is at this point that it becomes clear whether the proposed approach provides a real engineering advantage, rather than merely improving individual metrics.

Table 6

Aggregated evaluation of the practical value of the proposed method

Parametr	Detection	Structural	Confirmability
Metric	F1	κ	P_conf
Proposed	0.854	0.702	0.791
Baseline	0.832	0.653	n/a
Gain (%)	2.64	7.50	n/a
p-value	0.048	0.031	n/a
Effect size	0.49	0.62	n/a
Interpretation	Improved detection	Better clustering	Empirical validity

The most important aspect of Table 6 is that none of the metrics exists in isolation. An increase in F1-score of 2.64% alone would not be a sufficient argument in favor of a new approach. However, it is accompanied by a 7.50% increase in κ and the introduction of the metric P_{conf} = 0.791, for which no direct baseline analogue exists at all. In practical terms, this implies three simultaneous effects. First, the system detects anomalous transitions more accurately. Second, it does so within a less mixed and more interpretable space of execution regimes. Third, it enables the separation of formally risky transitions from those that have real operational manifestations. It is precisely here that an effect emerges which cannot be observed from F1-score alone—namely, the reduction of alert fatigue, improved incident prioritization, and faster root cause localization. Thus, the practical value of the approach is determined not by a single improvement, but by a coordinated integration of three analytical levels.

Conclusions

The relevance of the study is driven by the need to improve the reliability of automatic software error detection under conditions of increasing complexity of operational systems, where the isolated use of either formal rules or machine learning methods does not provide a complete and interpretable description of anomalous behavior.

For the first time, this paper substantiates a method for the quantitative evaluation of the empirical confirmability of invariant-oriented formal signals based on their systematic alignment with operational anomalies in program execution. This makes it possible to formalize the applicability boundaries of invariant analysis, to treat confirmability as an independent criterion for evaluating the quality of error detection models, and to theoretically justify the necessity of integrating formal and machine learning levels within a unified information technology framework.

Empirical verification confirmed the effectiveness of the approach in both detection and structural dimensions. The F1-score increased to 0.854 compared to 0.832 for the closest multimodal baseline method, corresponding to a 2.64% improvement, while PR-AUC increased to 0.873, resulting in approximately a 5–7% reduction in false positive rates at the same level of recall. In the structural dimension, a consistent improvement was observed in $\kappa = 0.702$, PQ = 0.864, and EQ = 0.398, corresponding to an increase in agreement of approximately 7.5% and a reduction in entropy-based cluster mixing by 15.7%. At the same time, it was established that 81.4% of invariant violations have empirical confirmation in execution logs, while 18.6% remain unconfirmed, and approximately 11.7% of empirical anomalies are not covered by the invariant layer, which reflects the bidirectional incompleteness of the analysis.

The practical significance of the obtained results lies in improving the efficiency of operational monitoring through simultaneous enhancement of anomaly detection accuracy, reduction of false positives, and the introduction of a quantitative confirmability criterion that enables the distinction between formally risky and operationally significant deviations, reduces the burden on incident analysis, and shortens root cause identification time.

The limitations of the study are associated with the use of a single experimental dataset (LO2), which complicates the direct generalization of the obtained quantitative relationships to other architectural classes of software systems, as well as with the dependence of the results on the quality of invariant layer construction, which in complex and dynamic environments may not fully capture all relevant behavioral patterns.

Future research should focus on generalizing the obtained results to different types of software systems, automating the synthesis of invariants based on data, and developing adaptive mechanisms for evaluating empirical confirmability that can account for the variability of operational environments and reduce both the proportion of unconfirmed formal signals and anomalies that remain outside the invariant-based description.

ADDITIONAL INFORMATION

Author Contributions

Conceptualization F.H.; methodology F.H.; validation F.H.; formal analysis, V.K.; investigation, F.H.; writing - original draft preparation, V.K.; writing - review and editing, F.H.; visualization, F.H., V.K.; project administration, V.K.

DECLARATION ON THE USE OF GENERATIVE ARTIFICIAL INTELLIGENCE TOOLS

In preparing this work, the author used Grammarly for: grammar and spelling checks. After using this tool/service, author reviewed and edited the content and take full responsibility for the content of this publication.

References

1. Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1285–1298). ACM. CCS '17: 2017 ACM SIGSAC Conference on Computer and Communications Security. https://doi.org/10.1145/3133956.3134015

2. Khlifi, M. K., Boulila, W., & Farah, I. R. (2023). Graph-based deep learning techniques for remote sensing applications: Techniques, taxonomy, and applications — A comprehensive review. Computer Science Review, 50, 100596. https://doi.org/10.1016/j.cosrev.2023.100596

3. Han, P., Li, H., Xue, G., & Zhang, C. (2023). Distributed system anomaly detection using deep learning-based log analysis. Computational Intelligence, 39(3), 433–455. https://doi.org/10.1111/coin.12573

4. Hashemi, S., & Mäntylä, M. (2022). SiaLog: detecting anomalies in software execution logs using the siamese network. Automated Software Engineering, 29(2). https://doi.org/10.1007/s10515-022-00365-7

5. Xie, Y., Zhang, H., & Babar, M. A. (2024). LogSD: Detecting Anomalies from System Logs through Self-Supervised Learning and Frequency-Based Masking. Proceedings of the ACM on Software Engineering, 1(FSE), 2098–2120. https://doi.org/10.1145/3660800

6. Duan, Y., Xue, K., Sun, H., Bao, H., Wei, Y., You, Z., Zhang, Y., Jiang, X., Yang, S., Chen, J., Duan, B., & Ou, Z. (2024). LogEDL: Log Anomaly Detection via Evidential Deep Learning. Applied Sciences, 14(16), 7055. https://doi.org/10.3390/app14167055

7. Audibert, J., Michiardi, P., Guyard, F., Marti, S., & Zuluaga, M. A. (2020). USAD. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 3395–3404). ACM. KDD '20: The 26th ACM

SIGKDD Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/3394486.3403392

8. Munir, M., Siddiqui, S. A., Dengel, A., & Ahmed, S. (2019). DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. IEEE Access, 7, 1991–2005. https://doi.org/10.1109/access.2018.2886457

9. He, J., Cheng, Z., & Guo, B. (2024). Anomaly detection in telemetry data using a jointly optimal one-class support vector machine with dictionary learning. Reliability Engineering & System Safety, 242, 109717. https://doi.org/10.1016/j.ress.2023.109717

10. Audibert, J., Michiardi, P., Guyard, F., Marti, S., & Zuluaga, M. A. (2022). Do deep neural networks contribute to multivariate time series anomaly detection? Pattern Recognition, 132, 108945. https://doi.org/10.1016/j.patcog.2022.108945

11. Zhao, N., Chen, J., Yu, Z., Wang, H., Li, J., Qiu, B., Xu, H., Zhang, W., Sui, K., & Pei, D. (2021). Identifying bad software changes via multimodal anomaly detection for online service systems. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 527–539). ACM. ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. https://doi.org/10.1145/3468264.3468543

12. Wang, P., Zhang, X., & Cao, Z. (2025). Anomaly detection for microservice system via augmented multimodal data and hybrid graph representations. Information Fusion, 118, 103017. https://doi.org/10.1016/j.inffus.2025.103017

13. Liu, W., Sun, D., Yang, H., Wang, Y., & Huang, W. (2026). Manod: A multi-modal anomaly detection framework for distributed system. Neural Networks, 193, 107999. https://doi.org/10.1016/j.neunet.2025.107999

14. Fan, M., Zhang, X., Wang, P., & Cao, Z. (2025). Multi-modal anomaly detection for microservice system through nested graph diffusion reconstruction. Applied Intelligence, 55(11). https://doi.org/10.1007/s10489-025-06681-1

Фредерік ГУРАЛЬНИК, В'ячеслав КОВТУН
Вінницький національний технічний університет

МЕТОД КІЛЬКІСНОГО ОЦІНЮВАННЯ ЕМПІРИЧНОЇ ПІДТВЕРДЖУВАНОСТИ ІНВАРІАНТНО-ОРІЄНТОВАНИХ СИГНАЛІВ У ЗАДАЧАХ АВТОМАТИЧНОГО ВИЯВЛЕННЯ ПОМИЛОК ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У статті розв'язано задачу підвищення достовірности автоматичного виявлення помилок у програмному забезпеченні на основі інтеграції формального інваріантного аналізу та методів машинного навчання. Дослідження зосереджено на проблемі розриву між інваріантно-орієнтованими формальними сигналами й емпірично спостережуваними аномаліями виконання програм, що обмежує ефективність як формальних, так і даних-орієнтованих підходів до аналізу логів і метрик. Уперше обґрунтовано метод кількісного оцінювання емпіричної підтверджуваности інваріантно-орієнтованих сигналів на основі їх системного зіставлення з експлуатаційними аномаліями виконання програм. Запропонований метод формалізує межі застосовности інваріантного аналізу, вводить підтверджуваність як самостійний критерій оцінювання якости моделей виявлення помилок і обґрунтовує необхідність інтеграції формального та машинно-навчального рівнів у межах єдиної інформаційної технології. Метод реалізовано через побудову переходів виконання як агрегованих поведінкових одиниць, їх мультимодальне подання за даними журналів і метрик та подальше зіставлення формальних і емпіричних сигналів у спільному аналітичному просторі. Емпіричну верифікацію виконано на датасеті LO2, що репрезентує мікросервісне середовище з журналами виконання, метриками та мітками коректних і помилкових станів. Запропонований підхід забезпечив значення гармонійної міри якости на рівні 0.854 і площі під кривою точности–повноти на рівні 0.873, а також поліпшення структурних характеристик моделі, зокрема коефіцієнта узгоджености до 0.702 та зниження ентропійної змішаности до 0.398. Встановлено, що 81.4% порушень інваріантів мають емпіричне підтвердження в логах виконання, тоді як 18.6% залишаються непідтвердженими. Це кількісно окреслює межу ефективности формального аналізу.

Ключові слова: виявлення аномалій; журнали виконання; експлуатаційні метрики; інваріанти програм; емпірична підтверджуваність; мультимодальний аналіз; машинне навчання; формальний аналіз.