https://doi.org/10.31891/csit-2026-2-15

UDC 004.4

MULTI-DRIFT PREDICTIVE MONITORING FOR EVOLVING INFORMATION SYSTEMS

Vasyl LYASHKEVYCH
Candidate of Technical Science, PhD, Associate Professor
Ivan Franko National University of Lviv, Lviv, Ukraine
https://orcid.org/0000-0003-2810-6061
e-mail: vasyl.liashkevych@lnu.edu.ua

This article addresses predictive monitoring of information systems under conditions of multidimensional functional-state evolution. Unlike conventional monitoring approaches focused on isolated anomalies, failures, or statistical deviations in data streams, the proposed approach treats an information system as a multilayer dynamic object influenced by interacting drift processes. The study considers nine drift types relevant to modern software-intensive and cyberinfrastructure environments: configuration, topology, role, policy, architectural, contextual, semantic, goal, and security drift. It is shown that these drifts affect not only current system parameters but also the validity of monitoring, interpretation, and decision-making processes. The current state of the field is analyzed and the literature is shown to remain fragmented across concept drift detection, multivariate change detection, software architecture erosion analysis, ontology evolution, role and policy evolution, context-aware access control, and self-adaptive systems. To address this fragmentation, the paper proposes an integrated predictive monitoring model based on an extended system state vector and a Predictive Drift Index for early identification of hazardous evolution trajectories. The model combines statistical, multivariate, architectural, contextual, semantic, and security-aware perspectives within a unified framework. A validation protocol is proposed, together with a simulation experiment based on controlled injection of isolated and combined drifts into a nine-dimensional system-state representation. The simulation demonstrates that the integrated predictive index reacts more clearly to multi-drift escalation than isolated indicators and supports earlier identification of degraded, vulnerable, anomalous, and critical trajectories. The proposed approach provides a basis for intelligent monitoring of evolving information systems.

Keywords: multi-drift monitoring, predictive monitoring, information systems, system evolution, machine learning, multi-agent systems, context-aware monitoring, adaptive monitoring

Introduction

Modern information systems (ISs) operate in an environment of continuous change: configurations, component interaction topology, user and agent roles, access rules, architectural solutions, execution context, data and knowledge semantics, system goals, and threat profiles are constantly changing. Therefore, traditional monitoring, which mostly captures current deviations in metrics, increasingly fails to reflect the actual dynamics of the IS. Instead, there is a growing need for predictive monitoring (PM), capable of detecting not only the fact of deviation, but also the trajectory of system-state drift before the IS enters degraded, vulnerable, anomalous, or critical modes [1-5].

The fundamental basis of this direction is the study of concept drift and change-point detection, which showed that data, models, and operating conditions in real ISs are not stationary [1-4, 6-8]. Further development of multidimensional monitoring has demonstrated that when moving to high-dimensional streams, new difficulties arise: changes are harder to detect, signs of drift may be distributed across many parameters, and local stability of individual metrics does not guarantee the stability of the IS as a whole [9-11]. For software-intensive ISs, this is especially important, since the software functional state (SFS) is not reduced to a single indicator but covers a set of characteristics related to quality, security, reliability, performance, environment, resource constraints, and software development life cycle (SDLC) context [5].

In ISs, drift can occur at different layers. Configuration drift manifests itself as a difference between the reference and

Received: 08/04/2026
Accepted: 06/05/2026
Published: 31/05/2026

actual configuration. Topology drift – as a change in the structure of connections between nodes, services, or agents. Role drift – as the evolution of roles and authorities. Policy drift – as an unauthorized or unnoticed deviation from access and management rules. Architectural drift – as a difference between the designed and the actually implemented architecture. Contextual drift – as a change in the operational environment. Semantic drift – as a change in the meaning of concepts and relationships in knowledge structures. Goal drift – as the evolution of goals and adaptation criteria. Security drift – as a change in the profile of threats, attacks, or protective assumptions [5, 12-27].

The relevance of each of these drifts is determined by the fact that it changes not only individual metrics, but also the conditions for correct monitoring. For example,

- a change in architecture or topology makes previous data collection routes and dependencies between components obsolete;
- a change in roles and policies changes permissible behavior patterns;
- a change in context or goals can make previous thresholds and quality criteria inadequate;
- a change in knowledge semantics distorts the interpretation of events;
- a change in the security profile changes the risk structure itself.

Research on context extraction methods in sustainable workplaces confirms that context can no longer be treated as an external auxiliary factor - it is a separate source of change and should be included in the PM model along with configuration, architectural and security parameters [26]. At the same time, works on the SFS and vulnerability as its key feature show that security and vulnerability should be considered not as an isolated incident class, but as an integral component of the SFS throughout the SDLC [5, 27].

This problem becomes even more significant in ISs developed using generative artificial intelligence, where recent research has highlighted the need for a unified architectural metamodel to reduce inconsistency, redundancy, and fragmentation of architectural artifacts, thereby increasing the relevance of monitoring for architectural drift in AI-driven software development [28].

Thus, the scientific problem lies in the lack of a generalized approach to PM of ISs in conditions of multi-drift, when changes simultaneously cover the configuration, topology, role, policy, architectural, contextual, semantic, goal, and security layers of the IS. The solution of this problem is directly related to the current scientific and practical tasks of ensuring functional stability, availability, security, adaptability, explainability of monitoring solutions and timely identification of critical trajectories of ISs development throughout their SDLC [1-25, 27, 29].

The remainder of the article is organized as follows. First, the current state of research on PM and drift-aware system analysis is reviewed. Next, known classes of drift models are compared, and a generalized PM model and method are proposed. Then a validation protocol is presented and an illustrative simulation experiment, followed by conclusions and directions for future work.

Related Work

In the broadest sense, the original core for building multi-drift models is the work on concept drift adaptation and learning under concept drift, where drift is interpreted as a change in the statistical properties of the data stream or the dependence between the input and target variables [1-2]. These works laid the foundation for detecting, explaining and adapting drift, but they mainly focused on ML models, and not on the IS itself as a multi-layer monitoring object. A close direction is represented by reviews of change-point detection and the taxonomy of model degradation, which show that changes can be sudden, gradual, repetitive, hidden or associated with a change in mode [3-4].

The next important class is online drift models, which are focused on the operational detection of changes in data streams. These include error-based detectors such as the Drift Detection Method (DDM) [6], adaptive window-based approaches such as ADaptive WINdowing (ADWIN) [7], and Bayesian online change-point detection [8]. Their advantage is early warning and suitability for streaming data, but they usually work either with model error or with a single aggregated signal, which means they do not capture structural, semantic, and goal shifts in the IS without an additional layer of modeling.

A separate group is made up of multidimensional models of changes, which are closest to the concept of multiparameter drift of an IS. The works of Kuncheva et al. [9], Qahtan et al. [10] and Alippi et al. [11] have shown that for multidimensional flows it is advisable to use likelihood-based criteria, subspace projections and detectability loss analysis. This is where the key idea for our article appears: drift is not necessarily localized in a separate parameter, it can manifest itself as a change in correlations, subspace or the total state structure. For software engineering, this approach is well consistent with the formalization of the SFS as a multidimensional construct that changes throughout the SDLC [5].

A separate subfield has been formed for architectural drift and architectural evolution. A systematic review by Breivold et al. [12] considers architectural evolution as a foundation for software evolvability, and Li et al. [13] systematizes the phenomenon of architecture erosion. The work by Uzun and Tekinerdogan [14] moves from a general description of the problem to a practical drift analysis based on architecture views. Taken together, these works prove that architectural changes should be tracked not only post factum, but as a separate object of continuous monitoring.

For topology drift, there are studies on detecting changes in network structure, in particular a statistical approach to network topology change detection [15]. Although this direction is more often developed separately from

software observability, it is critically important for distributed ISs, multi-agent systems, cloud orchestration and service meshes, where changes in connectivity directly affect causality tracing and monitoring routes.

Regarding role drift and policy drift, the most relevant body of research is related to role engineering, the evolution of role-based access control (RBAC) models, and continuous auditing of access control behavior. Classic role engineering scenario studies [16], as well as studies on the evolution of role definitions [17], indicate that roles are inherently dynamic rather than static, as they change with system usage patterns, organizational processes, and access requirements. The Policy Difference Detection (P-DIFF) approach [18], in turn, was developed specifically for the early detection of unintended changes in access control policies based on access log analysis. Additional support for this line of research is provided by context-sensitive security research, in particular the review by Kayes et al. [19], which systematizes the influence of spatial, temporal, and environmental contexts on access decisions and security mechanisms.

The closest studies to goal drift and contextual drift are self-adaptive systems, runtime goal models, and runtime uncertainty. Bencomo and Belaggoun [20] combine goal models with dynamic decision networks. Qureshi et al. [21] formulate requirements engineering for self-adaptive systems. de Lemos et al. [22] set a research roadmap for self-adaptive software. Gheibi and Weyns [23] directly talk about the drift of adaptation spaces. Giese et al. [24] interpret runtime models as a dynamic knowledge base about the system, the context, and the degree of satisfaction of the needs of stakeholders. An important addition to this direction is the work of Yakubovych and Lyashkevych [26], which proposed a method for obtaining context in sustainable workplaces. This work reinforces the thesis that contextual drift should be modeled through purposeful obtaining, structuring, and updating of context, and not only through the fixation of external changes.

Regarding semantic drift, the most representative studies are those on the evolution of ontologies. The work of Gulla et al. [25] introduces the concept of semantic drift in ontologies and proposes a way to detect it using conceptual signatures, while SemaDrift [29] develops a hybrid toolkit to measure semantic changes between ontology versions. For ISs based on knowledge graphs, rules, or ontologically driven reasoning, this means that semantic changes should be included in the monitoring cycle along with performance or security.

Ukrainian researchers have also made significant contributions to this field, particularly in three areas: network and Internet-of-Things monitoring, intelligent decision-support systems for forecasting, and conceptualization of SFS. One of the most important studies is the work of A. V. Lutsyuk, which uses specialized machine learning models for PM of information and communication systems and aims to predict potential network problems before they occur. The study reports a prediction accuracy of approximately 92% on the LUFlow dataset [30]. Another important contribution is the work of N. Lutsiv, T. Maksym'yuk, M. Beshley, O. Lavriv and co-authors on deep semi-supervised anomaly detection in heterogeneous ISs [31]. Although the term “predictive monitoring” is not used in the title, the research concerns the same conceptual space, namely the early detection of anomalous states in complex ISs with incompletely labeled data. A further methodological line is represented by the work of P. Bidyuk et al. on intelligent decision support systems for forecasting nonlinear non-stationary processes [32]. An additional contribution of Ukrainian researchers to this field is the work of S. Kunitska and S. Holub, who proposed a multi-agent information monitoring system based on multi-level intelligent monitoring technologies and cloud technologies. Their study is particularly relevant from the point of view of agent-oriented organization of monitoring processes, as it explicitly introduces predictor agents together with classifier agents and identifiers as elements of intelligent monitoring support [33]. These studies are not limited to ISs as purely technical objects, but they develop forecasting tools under uncertainty that are methodologically close to PM. Finally, the formalization of the SFS throughout the SDLC [5, 27] provides a solid conceptual basis for PM of software-intensive ISs in a broader, SDLC-oriented sense.

Ultimately, for security drift, it is important not only to monitor attacks or policy changes, but also to consider vulnerability as a feature of the SFS of an IS. Lyashkevych et al. [27] showed that vulnerability is one of the main characteristics of SFS throughout the SDLC. In combination with the formalization of the SFS concept itself [5], this gives grounds to interpret security drift not as a change in an isolated security indicator, but as a change in the entire subspace of SFS associated with risk, vulnerability, stability and operational safety.

Thus, existing works provide convincing results within individual classes of drift, but leave open the problem of a unified PM framework that would simultaneously cover statistical, structural, role-related, policy-related, contextual, semantic, goal-related, and security-related changes in the IS [1–32]. This is what the generalized model proposed below aims to solve.

Research Gap and Objectives

Despite the significant amount of research on concept drift detection, multidimensional change analysis, software architecture evolution, ontology evolution, context-aware access control, and self-adaptive ISs, the current literature remains fragmented across research areas. Most studies focus on specific manifestations of change, such as statistical drift in data flows, deviations between planned and implemented architectures, changes in access control behavior, or semantic evolution in knowledge structures. These approaches are rarely integrated into a unified perspective of PM of ISs as multi-level dynamic entities.

Accordingly, several research gaps remain unresolved. First, there is still no generalized model that can capture the drift of configuration, topology, roles, policies, architecture, context, semantics, goals, and security within

a single forecasting and monitoring system. Second, existing approaches differ significantly in their assumptions, data sources, and analytical mechanisms, making it difficult to unify them into a single operational model. Third, most available methods focus on detecting isolated changes rather than predicting the evolution of the IS state under the joint influence of multiple drift processes. Fourth, insufficient attention has been paid to the relationship between software drift processes and the SFS in its SDLC, especially regarding vulnerability, context sensitivity, and adaptive decision support.

Therefore, the purpose of this article is to develop an integrated framework for PM of ISs under multi-drift conditions. To achieve this goal, the study:

1) systematizes the current state of research on PM under drift conditions;

2) determines the applicability and limitations of existing classes of drift models with respect to the major drift types;

3) proposes a generalized model of PM that integrates statistical, multidimensional, architectural, semantic, contextual, and goal-oriented methods into a single representation of the state of the IS.

Results and Discussion

Current state of research on predictive drift monitoring. To clarify the current state of this field, the reviewed studies were grouped according to the main research directions in PM of ISs under drift conditions. The comparison focuses on the maturity of each direction, its major scientific contributions, and the limitations that still hinder the establishment of an integrated multi-drift PM framework. Consequently, Table 1 summarizes the scattered but complementary research achievements in areas such as concept drift detection, multivariate change analysis, architectural and topology evolution, role and policy change monitoring, contextual and semantic drift analysis, and security-oriented monitoring.

Table 1.

Current state of research directions relevant to PM under drift conditions

Research direction	Representative studies	Well-developed aspects	Remaining weaknesses
Concept drift / streaming machine learning	[1-2, 4, 6-8]	Detection of statistical changes, online adaptation, early warning	Limited coverage of structural, semantic, and goal-related changes
Multivariate drift / change detection	[6, 10-12]	Multivariate analysis, handling of correlations, subspaces, and change scores; connection with SFS	Difficult interpretation of drift causes; detectability loss in high-dimensional data
Architectural evolution / erosion / drift	[13-15]	Analysis of deviations between intended and implemented architecture; drift analysis	Limited integration with streaming monitoring and forecasting
Topology change detection	[16]	Detection of structural changes in networks and connectivity graphs	Weak connection with application-level semantics and runtime goals
Role / Role-Based Access Control evolution	[17-18]	Formalization of roles, evolution of roles based on usage patterns	Insufficient linkage to forecasting the system-level consequences of role drift
Policy drift / access-control change monitoring	[19-20]	Early detection of unintended policy changes, context-aware access control	Policies are often analyzed in isolation from architecture, goals, and semantics
Goal / adaptation-space drift	[21-25]	Runtime goals, adaptation spaces, uncertainty-aware adaptation	Lack of formal indices of goal drift integrated with observability
Contextual drift / context acquisition	[20-21, 25, 27]	Context modeling, runtime context interpretation, context-aware decisions	Insufficient integration with multi-drift forecasting
Semantic drift / ontology evolution	[26, 29]	Methods for detecting concept changes, stability ranking, comparison of ontology versions	Insufficient integration with real-time monitoring of software and infrastructure
Security drift / vulnerability-oriented monitoring	[5, 19-20, 28]	Detection of changes in access rules, context-aware security constraints, interpretation of vulnerability as a feature of functional state	Security drift is mostly not treated as a separate multidimensional trajectory of system-state evolution

As shown in Table 1, this field is quite mature across various drift-related research areas, but remains relatively fragmented from a comprehensive PM perspective. The most mature directions include concept drift detection, multivariate change detection, and several architecture-, policy-, and security-oriented monitoring methods. Meanwhile, the biggest weakness in this field lies in the lack of a unified framework that can collectively represent statistical, structural, contextual, semantic, goal-related, and security-related changes as components of an evolving system state. This observation indicates a need for a general PM model that can integrate the advantages of existing methods from a common multivariate drift perspective.

Analysis of known drift models. To further clarify the methodological foundation of PM under multiple drift conditions, it is necessary to compare the main classes of drift models used in the current scientific literature. Such analysis in Table 2 helps determine which types of change existing methods can detect most effectively, what their analytical advantages are, and in what ways their limitations become crucial when ISs simultaneously undergo

configuration, structural, contextual, semantic, and security-related changes. This study analyzes the models under consideration from the perspective of their applicability to characterizing and predicting nine types of drift, rather than simply focusing on their ability to detect isolated statistical deviations.

The analysis in Table 2 shows that existing drift-model classes are complementary rather than interchangeable. Statistical and streaming models are highly sensitive to time biases and suitable for early warning, but they offer limited support in terms of structural, semantic, and goal-related explanations. Architecture-based and access control-based methods have stronger explanatory power, but they are often passive and poorly integrated with predictive analytics. Ontology- and runtime-oriented models are more expressive in terms of semantics, context, and adaptive logic, but they are less applicable to continuous online monitoring. Therefore, the results confirm that no single existing model is sufficient for comprehensive PM of ISs under multiple drift conditions. This highlights the necessity of constructing an integrated PM model that combines the advantages of various methods, including statistical detection, multivariate analysis, architectural consistency assessment, contextual interpretation, semantic evolution analysis, and security-aware state assessment.

Table 2.

Analysis of known drift-model classes

Drift model class	Approach	Strengths	Limitations	Covered drift types
Error-rate / warning-level	DDM [6], EDDM derivatives	Simple, fast, online	Dependent on model error; weak in explaining the cause	security, policy, partly contextual
Adaptive windowing	ADWIN [7]	Handles changes in process rate, does not require a fixed window	Limited structural interpretability	configuration, contextual, security
Bayesian change-point	BOCPD [8]	Effective in detecting regime changes and transition points	Requires model assumptions	topology, contextual, goal, security
Likelihood / divergence-based multivariate	[9]	Suitable for streaming multivariate data	Complexity of scaling and explanation	configuration, topology, contextual, security
Principal Component Analysis / subspace-based	[5, 10]	Captures changes in correlations and latent structures; well aligned with the SFS	May lose semantic interpretability	architectural, topology, configuration, contextual
Detectability-aware multivariate	[11]	Addresses the problem of detectability loss in high-dimensional data	Requires an additional localization layer	all high-dimensional drifts as a meta-problem
Runtime-model / goal-model based	[20-24, 27]	Explains changes in goals, context, and adaptation space; context updating	Less suitable for purely statistical streaming detection	goal, contextual, policy, architectural
Architecture-conformance / drift analysis	[12-14]	Effectively localizes deviations between intended and implemented structure	Mostly reactive rather than predictive	architectural, topology, configuration
Role-Based Access Control / policy evolution models	[16-19]	Effective for roles, access, and normative constraints	Usually isolated from semantics and performance metrics	role, policy, security, contextual
Ontology / semantic evolution models	[25-26]	Enable tracking of changes in meanings, relationships, and concept stability	Weakly connected to low-level telemetry	semantic, goal, policy
Vulnerability-oriented functional-state models	[27, 29]	Integrate security, vulnerability, and life cycle into a unified state-based framework	Require operationalization for online drift detection	security, architectural, contextual, goal

Proposed generalized PM model. In this study, PM is interpreted as a continuous intelligent process of observing, integrating, evaluating, and predicting the evolution of the functional state of an IS under the simultaneous influence of many drifts. Unlike traditional monitoring approaches that focus mainly on isolated deviations of metrics or events, the proposed model considers an IS as a multilayer dynamic object whose state changes under the joint action of configuration, topology, role, policy, architectural, contextual, semantic, goal, and security drift.

Accordingly, at time t, the IS is represented by an extended state vector:

$$S(t) = [C(t), T(t), R(t), P(t), A(t), X(t), M(t), G(t), Sec(t)], \quad (1)$$

where $C(t)$ denotes the configuration state, $T(t)$ - the topology state, $R(t)$ - the role state, $P(t)$ - the policy state, $A(t)$ - the architectural state, $X(t)$ - the contextual state, $M(t)$ - the semantic state, $G(t)$ - the goal state, and $Sec(t)$ - the security and vulnerability state.

This representation extends the idea of the SFS, explicitly including not only operational and structural properties, but also context, semantics, adaptation goals, and security-related characteristics into a single analytical space. In this formulation, PM does not work solely on the basis of individual indicators; instead, it assesses the temporal evolution of the state of the integrated IS.

The multiparameter drift of an IS over the forecast horizon τ is defined as:

$$\Delta S(t, \tau) = S(t + \tau) - S(t), \quad (2)$$

It reflects the overall state displacement in the multidimensional monitoring space. However, since drift may affect not only parameter values but also internal dependencies, structures, and interpretations, a more complete drift representation is introduced as:

$$D(t,\tau) = \langle \Delta S(t,\tau), \Delta E(t,\tau), \Delta G(t,\tau), \Delta O(t,\tau) \rangle, \quad (3)$$

where $\Delta E(t,\tau)$ characterizes changes in statistical dependencies and correlations, $\Delta G(t,\tau)$ characterizes changes in graph topology and structural relations, and $\Delta O(t,\tau)$ characterizes semantic or ontological shifts in the interpretation of system entities, events, and relations.

To operationalize PM, each drift type is evaluated by a partial drift function:

$$d_i(t) = f_i(S(t), H(t), K(t)), \quad i = 1, \dots, 9, \quad (4)$$

where $d_i(t)$ is the normalized indicator of the i-th drift type, $H(t)$ denotes historical observations, and $K(t)$ denotes contextual and knowledge-based information used for interpretation.

Thus, the model generates a drift indicator vector:

$$d(t) = [d_C(t), d_T(t), d_R(t), d_P(t), d_A(t), d_X(t), d_M(t), d_G(t), d_{Sec}(t)]. \quad (5)$$

In order to capture the combined impact of different drift sources, the partial indicators are fused into an integrated predictive drift estimate. For this purpose, the model introduces the Predictive Drift Index:

$$PDI(t) = \sum_{i=1}^{g} w_i d_i(t) + \lambda_1 I_{corr}(t) + \lambda_2 I_{struct}(t) + \lambda_3 I_{sem}(t), \quad (6)$$

where w_i are the drift importance weights, $\lambda_1 I_{corr}(t)$ captures changes in multidimensional correlation, $\lambda_2 I_{struct}(t)$ captures structural and topological inconsistencies, and $\lambda_3 I_{sem}(t)$ captures the effects of semantic inconsistency or ontology evolution.

This formulation allows the model to combine the strengths of streaming statistical detection, multivariate change analysis, architectural conformance analysis, contextual reasoning, semantic evolution tracking, and security-aware state assessment. In the generalized version of the model, three higher-order interaction coefficients are sufficient to capture cross-drift effects at the levels of statistical dependence, structural change, and semantic inconsistency. However, for enterprise architecture management programs, this formulation can be extended to a hierarchical weighting scheme, in which additional weights are introduced for individual types of drift, architectural layers, or inter-layer dependencies.

The PM task is then formulated as forecasting the future functional state of the IS:

$$\hat{S}(t+\tau) = F(S(t), d(t), PDI(t), H(t), K(t)), \quad (7)$$

where F is a prediction function that estimates the future state of the system based on the current state, drift indicators, integrated drift index, historical evolution, and contextual knowledge.

In practice, F can be implemented using change point models, probabilistic forecasting, sequential models, graphical reasoning, or hybrid knowledge-based approaches. To support decision making, the future state is mapped to a set of monitoring classes, for example:

$$\hat{S}(t+\tau) \to \{S_N, S_D, S_V, S_A, S_F, S_R, S_U\}, \quad (8)$$

where S_N is normal, S_D is degraded, S_V is vulnerable, S_A is anomalous, S_F is failure-critical, S_R is recovering, and S_U is uncertain. This classification enables the monitoring system not only to forecast drift but also to interpret its consequences for system operability, resilience, and security.

Thus, the updated generalized PM model consists of four interconnected analytical levels. The first level performs data, event, context, and knowledge collection. The second level estimates nine partial drifts. The third level combines them into an integrated drift representation and calculates a predictive drift index. The fourth level predicts the future functional state and generates recommendations for early warning, explanation, and adaptive response. Thus, PM is transformed from detecting isolated deviations into a unified basis for predicting the multidimensional evolution of an IS under multi-drift conditions.

The proposed PM model integrates the strengths of several established research areas and combines them into a single multi-drift framework. From error-based and adaptive online methods such as Drift Detection Method, ADaptive WINdowing, and Bayesian online change-point detection, it inherits the sensitivity to temporal deviations and early warning capability under changing operating conditions [6-8]. From multivariate, subspace-based, and detectability-aware approaches, it inherits the ability to deal with multidimensional system states, hidden correlations, and latent structural changes that cannot be identified using isolated metrics alone [9-11]. From architecture drift analysis, the model borrows the ability to localize structural causes of deviations and interpret them at the level of architectural compliance and evolution [12-14]. Furthermore, the proposed model incorporates the analytical advantages of role-based and policy-based approaches, which allow for the interpretation of deviations related to access control structures, permission changes, and regulatory inconsistencies [16-19]. From runtime goal- and context-based models, it adopts a representation of adaptation goals, contextual conditions, and dynamic adaptation spaces, which is critical for ISs operating in non-stationary environments [20-24, 26].

From semantic-drift approaches, it inherits explanatory power at the level of knowledge structures, concept evolution, and meaning changes, which is especially important for ontological and knowledge-intensive ISs [25, 29]. Finally, by integrating the functional state perspective and vulnerability-oriented interpretation, the proposed model extends PM to the assessment of security-related degradation and critical system evolution taking into account the life cycle [5, 27].

From a scientific perspective, this allows the study to move beyond the narrow task of detecting drift in data streams toward PM of the evolution of an IS as a multilayer dynamic object. In this formulation, the nine types of drift are interpreted as interconnected coordinates of a single state space representation, rather than as isolated categories of deviations. This formulation provides a stronger foundation for intelligent monitoring, runtime reasoning, adaptive decision support, and future extensions towards monitoring environments focused on enterprise architecture management.

Analytical comparison of the proposed PM model. Our model (Table 3) does not deny existing approaches, but integrates them into a single state framework. Unlike the classical models of the family, which currently cover only the statistical, temporal or structural aspect of drift, the proposed model combines these perspectives within the multilayer state vector of the IS. That is why it is closer to the task of PM of multiparameter drift than any separate known family of models.

Table 3.

Analytical comparison of the proposed PM model with known drift-model families

Family of drift models	Core idea	Typical representatives	Strengths	Limitations	Relation to our model
Probabilistic-distributional models	Drift is treated as a change in data distributions or dependencies	Concept drift, covariate shift, prior/ posterior shift	Strong theoretical basis, suitable for ML monitoring	Weak structural and semantic interpretation	In our model for statistical drift estimation and baseline distribution monitoring
Change-point / regime-switching models	Drift is interpreted as transition between operational regimes or state segments	BOCPD, online/offline change-point detection	Good for transition detection and forecasting of state changes	Often depends on simplified assumptions about state transitions	Used in our model for temporal segmentation and regime transition forecasting
Subspace and latent-space models	Drift appears as a change in correlations, latent structure, or representation space	PCA-based, subspace-based, latent-space models	Suitable for high-dimensional system states	Lower explainability at business or semantic level	Used in our model for hidden multivariate state change analysis
Distance-based distribution models	Drift is measured as divergence between reference and current windows	KL, JS, Hellinger, Wasserstein, MMD	Useful in unsupervised monitoring	Sensitive to window design and feature representation	Used in our model for partial drift scoring
Density-ratio / density-difference models	Drift is estimated via density ratio or density difference	LSDD-type approaches	Efficient in online stream settings	Harder to interpret for architecture or policy drift	Can be used in our model for fine-grained statistical detectors
Statistical process control models	Drift is treated as a deviation from control boundaries	CUSUM, EWMA, Page–Hinkley	Fast online detection and early warning	Limited semantic and structural expressiveness	Used in our model for threshold-based detection of abrupt deviations
Error-rate / model-performance models	Drift is inferred from degradation of predictive accuracy or model behavior	DDM, EDDM, ADWIN-like approaches	Practical for predictive systems and adaptive ML	Detects consequences rather than the root cause	Used in our model as operational warning mechanisms
Hybrid closed-loop technical-system models	Drift detection is integrated with adaptation and control loops	CPS/cloud/observability frameworks	Strong practical value for dynamic technical systems	Often domain-specific and weakly generalized	Our model extends this class by adding semantic, policy, goal, and lifecycle dimensions
Proposed multi-drift PM model	Drift is interpreted as multidimensional evolution of the integrated state	Proposed model	Integrates 9 drift types	More complex calibration and validation	Serves as a unifying PM framework for evolving ISs

Comparative results show that the proposed model is not a replacement for existing drift-model classes, but rather integrates their analytical advantages into a general PM framework. Its main contribution lies in shifting the focus from isolated drift detection to predicting the state evolution of multidimensional systems. Therefore, this method is particularly suitable for ISs whose behavior is simultaneously affected by structural, contextual, semantic, policy, and security-related changes. Furthermore, this model remains scalable and can be further applied to enterprise architecture management, lifecycle-oriented monitoring, or intelligent decision support environments.

Advantages of the proposed model:

- Simultaneously covers nine types of drift instead of analyzing only one isolated type of change.
- Presents the IS as a multi-level dynamic object, and not just as a source of individual metrics or events.
- Combines numerical, structural, contextual, semantic and security characteristics in a single analytical framework.
- Supports the transition from simple drift detection to predicting the future functional state of the IS.

- Allows hidden correlations and inter-drift effects that are not visible when analyzing individual parameters to be taken into account.
- Provides better explainability of the results by decomposing the integral state into partial drifts.
- Is suitable for early warning of dangerous trajectories of system development.
- Supports generation of recommendations for adaptive response, including configuration, architectural, policy-related and security-related changes.
- Integrates the advantages of statistical, multidimensional, architectural, context-oriented and semantic approaches.
- Is consistent with the concept of the SFS throughout the SDLC.
- Allows security and vulnerability to be treated as components of the overall state of the system, rather than as a separate isolated class of events.
- Is suitable for further expansion in the areas of enterprise architecture management, runtime governance and intelligent decision support.

Model limitations:

- The proposed model is more complex than traditional single-level drift detectors, as it combines nine types of drift within a single state space.
- For correct operation, it requires a clear distinction between similar types of drift, in particular between architectural, topology, configuration and policy drift.
- Forecasting requires the availability of historical data suitable for calibrating partial indicators and the integrated index.
- The quality of the explanation of semantic drift and goal drift depends significantly on the completeness and quality of domain knowledge, ontologies or rules.
- Practical implementation requires the integration of heterogeneous data sources, in particular logs, metrics, traces, configurations, contextual and semantic resources; in a high-dimensional state space, the model may require additional dimensionality reduction or hierarchical decomposition.
- Security and vulnerability indicators may be difficult to unify across different organizations and classes of ISs.
- The model requires domain-specific customization for different stages of the SDLC.
- For full-fledged application in enterprise architecture management, additional formalization of architectural layers and inter-layer dependencies is required.

Multi-drift PM method. Table 2 shows that each class of drift models captures only a part of the overall dynamics of an evolving IS. Statistical and streaming methods are suitable for early drift detection, multivariate models are suitable for identifying hidden correlations and potential structural changes, while architectural, contextual, semantic, and security methods offer stronger explanatory power and domain-specific interpretability. However, in environments where multiple drift types coexist and influence each other, using these model categories in isolation is insufficient for PM. Therefore, the results demonstrate the rationale for transitioning from comparative analysis of existing models to developing a generalized PM model. In this model, configuration, topology, role, policy, architectural, contextual, semantic, goal, and security drift are considered as interrelated components of a unified system state vector. Based on this, the PM method (Fig. 1) is interpreted not as isolated drift detection, but as prediction of the evolution of the integrated functional state of the IS under multiple drift conditions.

Thus, the results of the analysis provide grounds to move from a qualitative description of a multi-drift environment to the formalization of a PM model. Such formalization is necessary to represent an IS as a dynamic multi-parameter object, the state of which changes under the influence of nine interconnected types of drift. Within the framework of this formulation, the PM model should determine the extended vector of the system state, the functions of partial drifts, the mechanism of their integration, as well as the rule for predicting the future functional state based on current observations, the history of changes, and the context of functioning.

Validation protocol. To ensure the reliability of the proposed model and PM method, the validation procedure should be organized as a step-by-step protocol covering formal, analytical, predictive, and practical aspects of the assessment.

Step 1. Build a data set and validation scenarios. The validation corpus should be formed from logs, metrics, traces, configuration snapshots, topology states, role and policy events, context signals, semantic resources, and security-related events. Based on this, a set of annotated scenarios should be prepared that include both isolated and combined manifestations of configuration, topology, role, policy, architecture, context, semantics, goal, and security drift. Each scenario should contain reference drift labels, drift onset points, and expected future functional state.

Step 2. Partial drift detection verification. In this step, each drift-specific detector is evaluated independently. The goal is to determine whether the method correctly identifies the presence, type, and onset of each drift. The result of this step is a set of partial drift indicators along with their detection quality metrics.

Step 3. Validation of the integrated multi-drift estimate. The partial drift metrics are then combined into an integrated drift representation and a predictive drift index. At this stage, validation assesses whether the combined model correctly accounts for drift interactions, hidden dependencies, and multifactor system degradation. The combined results are compared with expert annotations and real scenario data.

Step 4. Checking the predictive ability. The predictive component of the method is tested on time scenarios to assess whether the future functional state of the IS is correctly predicted over the selected horizon. The predicted state classes or predictive drift values are compared with the actually observed states after the predicted interval.

Step 5. Validation of decision support utility. Finally, the practical value of the proposed method is assessed by comparing it with conventional monitoring. The comparison should focus on whether PM provides earlier warnings, clearer explanations of dominant drifts, and more useful recommendations for adaptation to prevent degraded, vulnerable, or critical states.

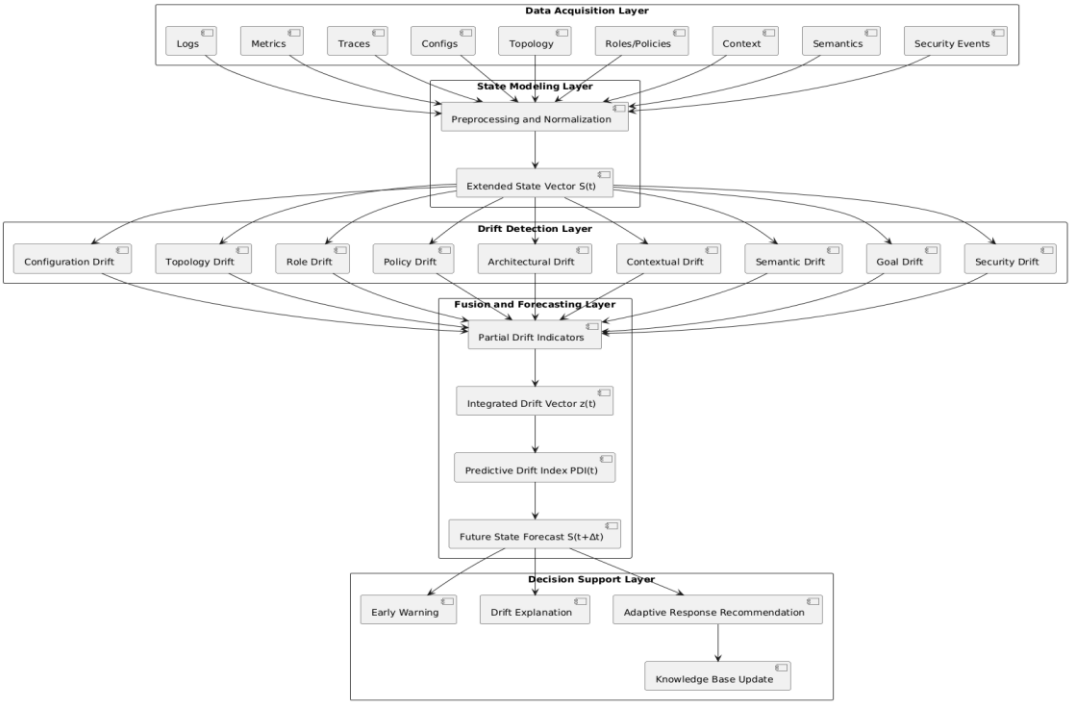

Fig. 1. Schema of multi-drift PM method

The validation of the proposed PM method requires a differentiated set of metrics, since the considered drift types affect different layers of the IS and therefore cannot be evaluated adequately by a single universal indicator. For this reason, Table 4 summarizes the main validation goals and the most appropriate metrics for each drift type, together with additional interpretation criteria relevant for integrated multi-drift assessment.

Table 4.

Validation metrics by drift type

Drift type	Main validation goal	Recommended metrics	Additional interpretation criteria
Configuration drift	Correct detection of deviations between expected and actual configuration states	Precision, Recall, F1-score, Detection Delay	Severity of deviation, reproducibility across environments
Topology drift	Correct detection of structural changes in component or network relations	Precision, Recall, F1-score, Structural Change Accuracy, Detection Delay	Graph inconsistency level, affected connectivity scope
Role drift	Correct identification of evolving role definitions and role-usage deviations	Precision, Recall, F1-score	Role-assignment instability, change impact on access behavior
Policy drift	Correct detection of unintended access-control or governance-policy changes	Precision, Recall, F1-score, False Positive Rate	Policy inconsistency level, compliance deviation
Architectural drift	Correct detection of divergence between intended and implemented architecture	Precision, Recall, F1-score, Drift Localization Accuracy	Number of affected components/views, architectural impact level
Contextual drift	Correct identification of changes in environmental or operational context	Precision, Recall, F1-score, Detection Delay	Context sensitivity, adaptability to non-stationary conditions
Semantic drift	Correct detection of concept, relation, or ontology evolution affecting interpretation	Precision, Recall, F1-score	Semantic inconsistency degree, ontology stability change
Goal drift	Correct identification of changes in adaptation goals or system objectives	Precision, Recall, F1-score	Goal misalignment degree, adaptation-space deviation
Security drift	Correct detection of shifts in vulnerability, threat profile, or security state	Precision, Recall, F1-score, False Negative Rate, Detection Delay	Risk escalation level, vulnerability-state impact
Integrated multi-drift state	Correct fusion of partial drifts into global PM result	Macro-F1, Accuracy, MAE/RMSE for PDI, Early Warning Lead Time	Explainability, robustness, cross-drift sensitivity

The validation protocol of the proposed PM method consists of five stages: construction of annotated scenarios for single and multiple drifts; validation of partial drift detection; validation of integrated multi-drift assessment; validation of future state prediction; and validation of decision support utility compared to conventional monitoring. Such a protocol allows evaluating the method at the levels of detection accuracy, prediction quality, reliability, and practical applicability.

Simulation experiment. To provide an initial validation of the proposed PM model, a controlled simulation experiment was designed in which an IS was represented as a nine-dimensional dynamic state vector (1), where the components correspond to configuration, topology, role, policy, architecture, contextual, semantic, goal, and security states, respectively. Each component was normalized to the interval $[0,1]$, where higher values indicate greater deviation from the nominal operating mode.

The baseline behavior of the simulated IS was generated by a low-noise autoregressive process in discrete time $t = 1,\dots,N$. This ensures smooth system evolution without drift. Controlled drift signals are injected at predefined time intervals above this baseline. The simulation includes isolated and combined drift scenarios to reproduce simple and complex system evolution patterns. Specifically, configuration drift is defined (Table 5) as abrupt deviation, topology drift as asymptotic structural change, role and policy drift as concurrent access-related change, architectural drift as cumulative gradual deviation, context drift as transient perturbation, semantic and goal drift as knowledge- and goal-related change, and security drift as a late-stage perturbation superimposed on accumulated system instability.

Table 5.

Drift-injection plan in the simulation experiment

Phase	Time interval	Injected drift	Type of injection	Intensity
1	1–19	none	baseline	0
2	20–34	configuration drift	abrupt	0.35
3	35–49	topology drift	gradual	0.30
4	50–64	role drift + policy drift	abrupt + gradual	0.20 + 0.25
5	65–79	architectural drift	gradual	0.35
6	80–89	contextual drift	transient	0.30
7	90–99	semantic drift + goal drift	gradual + abrupt	0.25 + 0.20
8	100–120	security drift + combined amplification	abrupt + joint effect	0.40

For each component, a partial drift index was calculated based on the deviation from the moving control window. These indices were then aggregated into an integrated predictive drift index that combines the weighted contributions of partial drift with higher-level interaction effects related to changes in multivariate correlation, structural mismatch, and semantic mismatch. The resulting index was used as the main input for predicting the future functional state of the IS.

The experiment was organized over 120 discrete time steps. During the first phase, the system evolved in a near-steady state and served as a reference baseline. Subsequent phases introduced different types of drift at predetermined times, allowing for a direct comparison between the introduced drift events and the observed drift responses. The time evolution of the nine simulated state components is presented in Fig. 2, which illustrates the transition from the baseline to progressively more unstable multi-drift behavior.

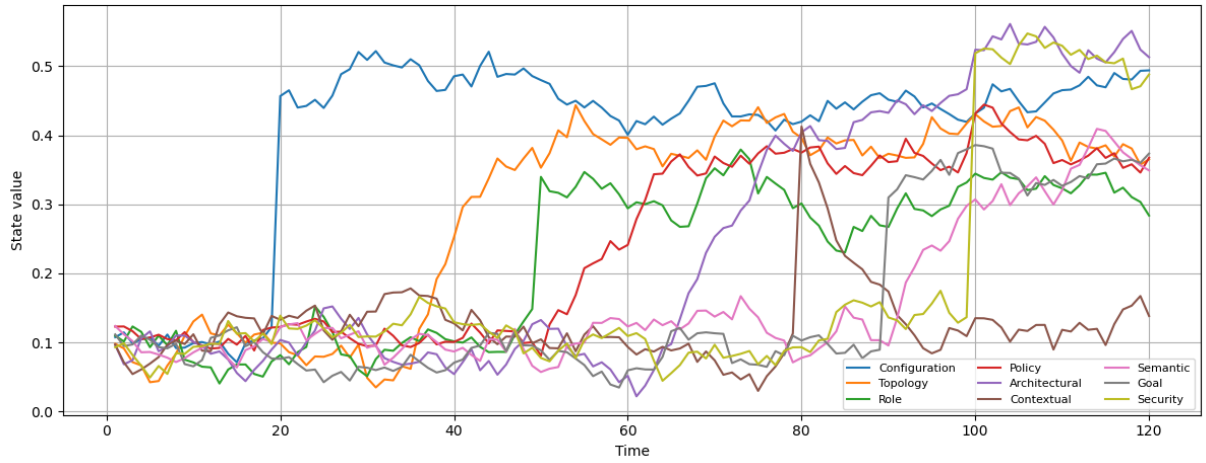

Fig. 2. Simulated temporal evolution of the nine state components of the IS under sequential and combined drift injections

The corresponding partial drift indicators are shown in Fig. 3, where the activation of individual drift-sensitive components can be observed at the moments of the introduced perturbations.

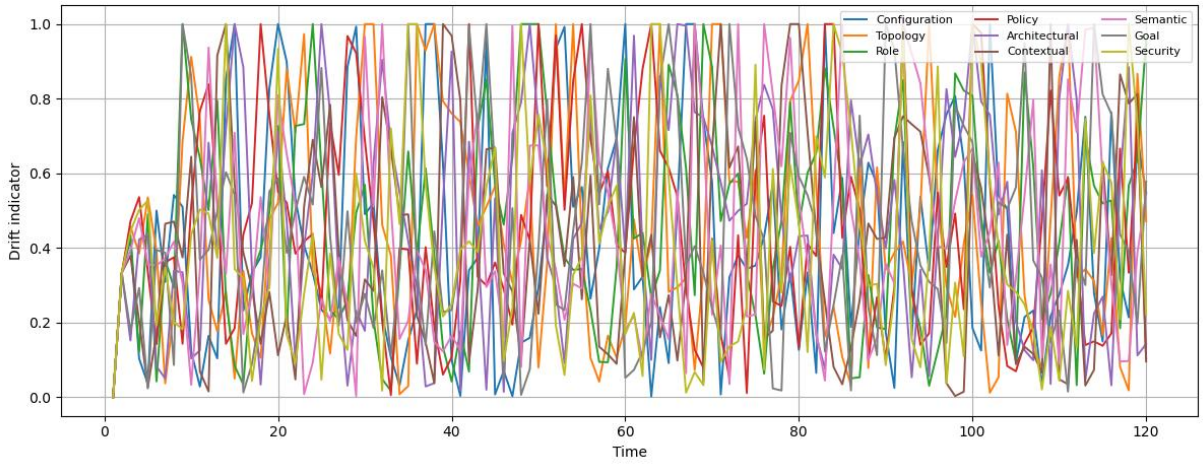

Fig. 3. Partial drift indicators corresponding to the nine monitored drift types in the simulation experiment

Finally, the integrated predictive drift index together with the predicted risk trajectory is presented in Fig. 4, demonstrating that the combined PM output increases more clearly under combined drift conditions and provides an earlier indication of the evolution of the hazardous system than changes at the level of individual components.

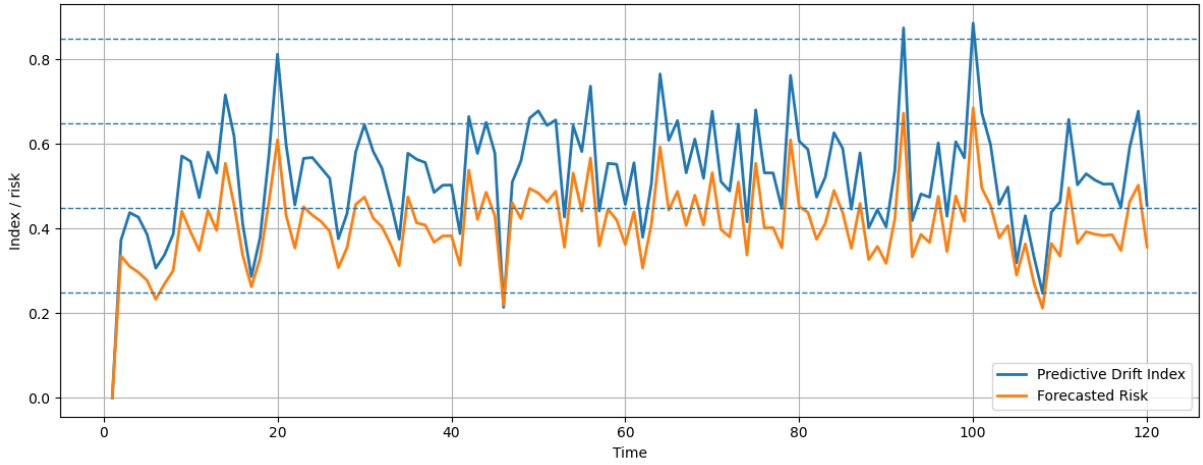

Fig. 4. Integrated Predictive Drift Index and forecasted risk trajectory under multi-drift conditions

The simulation results included a time series of state components, partial drift indicators, integrated predictive drift index, and class of predicted future state of the IS. Overall, the experiment allowed us to assess the sensitivity of the proposed method to different drift patterns, the consistency between the introduced and detected changes, and the ability of the integrated predictive index to provide early warning of degraded, vulnerable, anomalous, and critical IS trajectories.

Conclusions

This study addressed the problem of PM of evolving ISs in the context of multifactorial drift. The analysis showed that existing research provides significant results in individual areas related to drift, including conceptual drift detection, multidimensional change analysis, architectural drift analysis, ontology evolution, access control evolution, and self-adaptive systems, but remains fragmented in terms of integrated PM of ISs as multi-layered dynamic entities. Ukrainian research contributes to this field through PM of networks and the Internet of Things, semi-supervised anomaly detection in heterogeneous ISs, decision support systems for forecasting under uncertainty, and lifecycle-oriented formalization of the SFS [5, 27, 30–32].

To address this fragmentation, the paper proposes a generalized PM model in which an IS is represented by an extended state vector that includes configuration, topology, role, policy, architectural, contextual, semantic, goal, and security components. On this basis, multi-drift evolution is interpreted as a multidimensional state shift of the system, while PM is formulated as the continuous observation, aggregation, and prediction of drift-sensitive state changes. The proposed predictive drift index allows combining partial drift measures with higher-order interaction effects related to statistical dependence, structural inconsistency, and semantic inconsistency.

The scientific novelty of the study lies in:

- identifying the analytical fragmentation of existing approaches to PM of IS related to drift;
- proposing a generalized PM model that represents nine types of drift as interrelated dimensions of a single system state space;
- developing a multi-stage PM method that integrates drift-specific detection, drift aggregation, future state prediction, and adaptive response support;
- implementing a validation protocol that combines formal, analytical, predictive, and decision support evaluation for multi-drift monitoring;
- linking PM to SFS and vulnerability as characteristics of evolving, lifecycle-sensitive ISs.

The practical contribution of the work lies in the proposed validation protocol and initial simulation-based evaluation. A controlled simulation experiment demonstrated that the integrated PM output responds more clearly to the combined drift escalation than isolated component-level indicators and can provide early warning of degraded, vulnerable, anomalous, and critical system trajectories. Although the experiment is illustrative and does not replace full empirical validation on real monitoring data, it confirms the feasibility of the proposed framework as a basis for future implementation and testing.

The main limitations of the proposed approach are the complexity of integrating heterogeneous data sources, the need to distinguish between closely related types of drift, the dependence on historical data for calibration, and the need to operationalize semantic, goal, and security indicators focused on a specific subject area.

Therefore, future work should focus on large-scale empirical validation in real-world environments with intensive use of software and cyberinfrastructure, comparison of datasets for multi-drift predictive monitoring, layer-aware weighting schemes for enterprise architecture management, explainable hybrid implementations combining statistical, graph-based, and knowledge-driven models, and comparative evaluation with traditional monitoring methods based on single drift, anomalies, and threshold knowledge.

ADDITIONAL INFORMATION

DECLARATION ON THE USE OF GENERATIVE ARTIFICIAL INTELLIGENCE TOOLS

During the preparation of this work, the author used X-GPT-5.4 in order to: Grammar and spelling check. After using this tool /service, the author reviewed and edited the content as needed and took full responsibility for the publication’s content.

References

1. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A Survey on Concept Drift Adaptation. ACM Computing Surveys, 46(4), Article 44. https://doi.org/10.1145/2523813
2. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under Concept Drift: A Review. IEEE Transactions on Knowledge and Data Engineering, 31(12), 2346–2363. https://doi.org/10.1109/TKDE.2018.2876857
3. Aminikhanghahi, S., & Cook, D. J. (2017). A Survey of Methods for Time Series Change Point Detection. Knowledge and Information Systems, 51(2), 339–367. https://doi.org/10.1007/s10115-016-0987-z
4. Bayram, F., & Arslan, H. (2022). From concept drift to model degradation: An overview on performance-aware drift detectors. Knowledge-Based Systems, 245, 108632. https://doi.org/10.1016/j.knosys.2022.108632
5. Lyashkevych, M. Y., Lyashkevych, V. Y., & Shuvar, R. Y. (2025). Definition and Formalization of the Software Functional State Concept Throughout the Development Life Cycle. Electronics and Information Technologies, 32, 151–170. https://doi.org/10.30970/eli.32.11
6. Gama, J., Medas, P., Castillo, G., Rodrigues, P. (2004). Learning with Drift Detection. In: Bazzan, A.L.C., Labidi, S. (eds) Advances in Artificial Intelligence – SBIA 2004. SBIA 2004. Lecture Notes in Computer Science(), vol 3171. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-28645-5_29
7. Bifet, A., & Gavaldà, R. (2007). Learning from Time-Changing Data with Adaptive Windowing. In Proceedings of the 7th SIAM International Conference on Data Mining (pp. 443–448). https://doi.org/10.1137/1.9781611972771.42
8. Adams, R. P., & MacKay, D. J. C. (2007). Bayesian Online Changepoint Detection. arXiv:0710.3742. https://doi.org/10.48550/arXiv.0710.3742
9. Kuncheva, L. I., & Faithfull, W. J. (2014). Change Detection in Streaming Multivariate Data Using Likelihood Detectors. IEEE Transactions on Knowledge and Data Engineering, 26(5), 1175–1180. https://doi.org/10.1109/TKDE.2011.226
10. Qahtan, A. A., Alharbi, B., Wang, S., & Zhang, X. (2015). A PCA-Based Change Detection Framework for Multidimensional Data Streams. In Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 935–944). https://doi.org/10.1145/2783258.2783359
11. Alippi, C., Boracchi, G., Carrera, D., & Roveri, M. (2016). Change Detection in Multivariate Datastreams: Likelihood and Detectability Loss. In

Proceedings of IJCAI 2016 (pp. 1368–1374). Derived from: https://www.ijcai.org/Proceedings/16/Papers/197.pdf

12. Breivold, H. P., Crnkovic, I., & Larsson, M. (2012). A systematic review of software architecture evolution research. Information and Software Technology, 54(1), 16–40. https://doi.org/10.1016/j.infsof.2011.06.002
13. Li, R., Liang, P., & Avgeriou, P. (2022). Understanding software architecture erosion: A systematic mapping study. Journal of Software: Evolution and Process, 34(6), e2423. https://doi.org/10.1002/smr.2423
14. Uzun, B., & Tekinerdogan, B. (2024). Detecting deviations in the code using architecture view-based drift analysis. Information Systems, 119, 103774. https://doi.org/10.1016/j.csi.2023.103774
15. Liu, Y., Liu, L., & Yan, Y. (2020). Network Topology Change Detection Based on Statistical Process Control. In Proceedings of HPCCT & BDAI 2020 (pp. 145–151). https://doi.org/10.1145/3409501.3409532
16. Neumann, G., & Strembeck, M. (2002). A Scenario-driven Role Engineering Process for Functional RBAC Roles. In Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (pp. 33–42). https://doi.org/10.1145/507711.507717
17. Zhang, W., Chen, Y., Gunter, C. A., Liebovitz, D., & Malin, B. (2013). Evolving role definitions through permission invocation patterns. In Proceedings of the 18th ACM Symposium on Access Control Models and Technologies (pp. 37–48). https://doi.org/10.1145/2462410.2462422
18. Xiang, C., Wu, Y., Shen, B., Shen, M., Huang, H., Xu, T., Zhou, Y., Moore, C., Jin, X., & Sheng, T. (2019). Towards Continuous Access Control Validation and Forensics. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (pp. 113–129). https://doi.org/10.1145/3319535.3363191
19. Kayes, A. S. M., Kalaria, R., Sarker, I. H., Islam, M. S., Watters, P. A., Ng, A., Hammoudeh, M., Badsha, S., & Kumara, I. (2020). A Survey of Context-Aware Access Control Mechanisms for Cloud and Fog Networks: Taxonomy and Open Research Issues. Sensors, 20(9), 2464. https://doi.org/10.3390/s20092464
20. Bencomo, N., & Belaggoun, A. (2013). Supporting Decision-Making for Self-Adaptive Systems: From Goal Models to Dynamic Decision Networks. In REFSQ 2013 (LNCS 7830, pp. 221–236). Springer. https://doi.org/10.1007/978-3-642-37422-7_16
21. Qureshi, N.A., Jureta, I.J., Perini, A. (2011). Requirements Engineering for Self-Adaptive Systems: Core Ontology and Problem Statement. In: Mouratidis, H., Rolland, C. (eds) Advanced Information Systems Engineering. CAiSE 2011. Lecture Notes in Computer Science, vol 6741. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21640-4_5
22. de Lemos, R. et al. (2013). Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds) Software Engineering for Self-Adaptive Systems II. Lecture Notes in Computer Science, vol 7475. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35813-5_1
23. Gheibi, O., & Weyns, D. (2024). Dealing with Drift of Adaptation Spaces in Learning-based Self-Adaptive Systems Using Lifelong Self-Adaptation. ACM Transactions on Autonomous and Adaptive Systems, 19(1), Article 2. https://doi.org/10.1145/3636428
24. Giese, H. et al. (2014). Living with Uncertainty in the Age of Runtime Models. In: Bencomo, N., France, R., Cheng, B.H.C., Aßmann, U. (eds) Models@run.time. Lecture Notes in Computer Science, vol 8378. Springer, Cham. https://doi.org/10.1007/978-3-319-08915-7_3
25. Gulla, J. A., Solskinnsbakk, G., Myrseth, P., Haderlein, V., & Cerrato, O. (2010). Semantic Drift in Ontologies. In Proceedings of WEBIST 2010 (pp. 13–20). https://doi.org/10.5220/0002788800130020
26. Yakubovych, M. I., & Lyashkevych, V. Y. (2026). Context obtaining method in sustainable workplaces. Informatics and Mathematical Methods in Simulation, 16(1), 34–42. https://doi.org/10.15276/imms.v16.no1.34
27. Lyashkevych, M. Y., Lyashkevych, V. Y., & Shuvar, R. Y. (2025). Vulnerability as a main feature of the functional state which characterises software security during the life cycle. Information Technology: Computer Science, Software Engineering and Cyber Security, 4, 146–155. https://doi.org/10.32782/IT/2025-4-17
28. Grynets, O., & Lyashkevych, V. (2026). Unified Architecture Metamodel of Information Systems Developed by Generative AI. arXiv. https://arxiv.org/abs/2604.00171
29. Stavropoulos, T. G., Vrakas, D., & Vlahavas, I. (2019). SemaDrift: A hybrid method and visual tools to measure semantic drift in ontologies. Web Semantics, 54, 1–15. https://doi.org/10.1016/j.websem.2018.05.001.
30. Lutsiuk, A. V. (2024). Predictive monitoring of information and communication systems using a specialized machine learning model. Scientific Notes of Tavria National University named after V. I. Vernadsky. Series: Technical Sciences. https://doi.org/10.32782/2663-5941/2024.6.1/22
31. Lutsiv, N., Maksymyuk, T., Beshley, M., Lavriv, O., Andrushchak, V., Sachenko, A., Vokorokos, L., & Gazda, J. (2022). Deep semisupervised learning-based network anomaly detection in heterogeneous information systems. Computers, Materials & Continua, 70(1), 413–431. https://doi.org/10.32604/cmc.2022.018773

32. Bidiuk, P., Prosyankina-Zharova, T., Diakon, V., & Diakon, D. (2023). The improvement of the intelligent decision support system for forecasting non-linear non-stationary processes. Technology Audit and Production Reserves, 4(2(72)), 37–46. https://doi.org/10.15587/2706-5448.2023.286516

33. Kunytska, S., & Holub, S. (2020). Multi-agent Monitoring Information Systems. In A. Palagin, A. Anisimov, A. Morozov, & S. Shkarlet (Eds.), Mathematical Modeling and Simulation of Systems (MODS 2019), Advances in Intelligent Systems and Computing, vol. 1019, pp. 164–171. Springer, Cham. https://doi.org/10.1007/978-3-030-25741-5_17

Василь ЛЯШКЕВИЧ
Львівський національний університет імені Івана Франка

БАГАТО-ДРЕЙФОВИЙ ПРОГНОЗНИЙ МОНІТОРИНГ ДЛЯ ІНФОРМАЦІЙНИХ СИСТЕМ, ЩО ЕВОЛЮЦІОНУЮТЬ

Ця стаття розглядає прогнозний моніторинг інформаційних систем в умовах багатовимірної еволюції функціональних станів. На відміну від традиційних підходів до моніторингу, зосереджених на ізольованих аномаліях, збоях або статистичних відхиленнях у потоках даних, запропонований підхід розглядає інформаційну систему як багатошаровий динамічний об'єкт, на який впливають взаємодіючі процеси дрейфу. У дослідженні розглядається дев'ять типів дрейфу, що стосуються сучасних програмно-інтенсивних та кіберінфраструктурних середовищ: конфігураційний, топологічний, рольовий, дрейф політик, архітектурний, контекстний, семантичний, дрейф цілей та безпековий. Показано, що ці дрейфи впливають не лише на поточні параметри системи, але й на достовірність процесів моніторингу, інтерпретації та прийняття рішень. Проаналізовано поточний стан галузі, і показано, що література залишається фрагментованою за такими напрямками, як виявлення дрейфу концепцій, виявлення багатовимірних змін, аналіз ерозії архітектури програмного забезпечення, еволюція онтологій, еволюція ролей та політик, контекстно-залежний контроль доступу та самоадаптивні системи. Щоб вирішити цю фрагментацію, у статті пропонується інтегрована модель прогнозного моніторингу, заснована на розширеному векторі стану системи та індексі прогнозного дрейфу для ранньої ідентифікації небезпечних траєкторій еволюції. Модель поєднує статистичні, багатовимірні, архітектурні, контекстні, семантичні та безпеково-орієнтовані аспекти в єдиній структурі. Запропоновано протокол валідації разом із симуляційним експериментом, заснованим на контрольованому впровадженні ізольованих та комбінованих дрейфів у дев'ятивимірне представлення стану системи. Імітаційний експеримент демонструє, що інтегрований індекс прогнозного дрейфу чіткіше реагує на ескалацію кількох дрейфів, ніж ізольовані індикатори, та підтримує більш ранню ідентифікацію деградованих, вразливих, аномальних та критичних траєкторій.

Ключові слова: багато-дрейфовий моніторинг, прогнозний моніторинг, інформаційні системи, еволюція системи, машинне навчання, багатоагентні системи, контекстно-залежний моніторинг, адаптивний моніторинг