УДК 519.2

Научная статья

https://doi.org/10.32603/2071-8985-2026-19-1-52-59

Математическая модель для измерения конфигурационного дрейфа в ИТ-инфраструктуре

К. О. Моисеев✉, А. О. Трофимова

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина), Санкт-Петербург, Россия

✉ vihuhool@vk.com

Аннотация. Предлагается математическая модель для количественной оценки конфигурационного дрейфа в ИТ-инфраструктуре. Каждая проверка конфигурации описывается нормированной шкалой соответствия, позволяющей учитывать как бинарные, так и градуированные параметры. Проверки группируются по непересекающимся категориям, что обеспечивает интерпретируемость и возможность декомпозиции результатов. Модель вводит иерархическую систему весов, позволяющую агрегировать оценки от уровня отдельных проверок до узлов, ролей и кластера в целом. Представлены формальные свойства индекса соответствия и индекса дрейфа, включая ограниченность, аддитивность и устойчивость к изменениям шкал. Приводятся рекомендации по формированию атомарных проверок и корректных шкал, а также демонстрационный пример вычисления индекса для сервера с несколькими категориями параметров. Предложенный подход формирует строгую теоретическую основу для последующего построения систем мониторинга и управления конфигурационным дрейфом.

Ключевые слова: конфигурационный дрейф, ИТ-инфраструктура, нормированная метрика, индекс соответствия, агрегирование, иерархические веса, шкалы измерения, математическая модель

Для цитирования: Моисеев К. О., Трофимова А. О. Математическая модель для измерения конфигурационного дрейфа в ИТ-инфраструктуре // Изв. СПбГЭТУ «ЛЭТИ». 2026. Т. 19, № 1. С. 52–59. doi: 10.32603/2071-8985-2026-19-1-52-59.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Original article

A Mathematical Model for Assessing Configuration Drift in it Systems

K. O. Moiseev✉, A. O. Trofimova

Saint Petersburg Electrotechnical University, Saint Petersburg, Russia

✉ vihuhool@vk.com

Abstract. This paper introduces a mathematical model for quantitatively assessing configuration drift in IT infrastructure. Each configuration check is represented by a normalized compliance scale, supporting both binary and graded parameters. Checks are grouped into non-overlapping categories to ensure interpretability and enable decomposition of results. The model defines a hierarchical system of weights that allows aggregation from individual checks to nodes, roles, and clusters. Formal properties of the compliance index and the drift index are established, including boundedness, additivity, and robustness to scale transformations. Guidelines are provided for designing atomic checks and consistent measurement scales, and a demonstration example illustrates the calculation of the drift index for a server with multiple parameter categories. The proposed approach provides a rigorous theoretical foundation for further development of monitoring and management systems for configuration drift.

Keywords: configuration drift, IT infrastructure, normalized metric, compliance index, aggregation, hierarchical weights, measurement scales, math model.

For citation: Moiseev K. O., Trofimova A. O. A Mathematical Model for Assessing Configuration Drift in it Systems // LETI Transactions on Electrical Engineering & Computer Science. 2026. Vol. 19, no. 1. P. 52–59. doi: 10.32603/2071-8985-2026-19-1-52-59.

Conflict of interest. The authors declare no conflicts of interest.

Введение. Информационные технологии (ИТ) лежат в основе современных крупных инфраструктур. В таких ИТ-инфраструктурах фактическое состояние серверов и других компонентов неизбежно со временем расходится с эталонной конфигурацией. На это влияют обновления версий пакетов, экстренные изменения, особенности порядка развертывания и устранения инцидентов. Эти изменения накапливаются и образуют дрейф конфигураций.

При незначительном дрейфе последствия остаются малозаметными. Однако по мере роста отклонений увеличивается вероятность сбоев, уязвимостей, нарушений требований безопасности и соответствия стандартам, а также возрастает риск невоспроизводимости сред и срыва сроков при развертывании новых версий систем.

Распространенный способ контроля заключается в проведении периодических проверок по заранее заданным правилам. Такой подход полезен для локального обнаружения несоответствий, но имеет серьезные ограничения:

– отсутствует единый показатель дрейфа, позволяющий оценивать текущее состояние в целом и сравнивать различные узлы и роли между собой;

– не учитывается различная значимость проверок, что затрудняет определение приоритетов;

– невозможно оценить динамику изменений и выявить нарастающие тенденции до наступления критических последствий.

В результате руководителям сложно расставлять приоритеты и планировать работы по исправлению, а инженерам – аргументированно обосновывать необходимость тех или иных изменений.

Для решения этих проблем необходима модель, которая позволит переходить от перечня отдельных нарушений к интегральной числовой оценке состояния конфигураций и обеспечит сопоставимость результатов на разных уровнях – от отдельных узлов до всей ИТ-инфраструктуры.

Анализ существующих подходов. Современные инженерные практики и исследования в области управления конфигурациями в ИТ-инфраструктурах сосредоточены преимущественно на автоматизации выявления и устранения отклонений между фактическим и эталонным состояниями. В парадигме «инфраструктура-как-код» (Infrastructure as Code, IaC) конфигурационный дрейф рассматривается как естественное следствие динамики изменений, а его устранение реализуется через периодические проверки и автоматическое приведение системы к эталону. Инструменты управления конфигурациями – Ansible, Puppet, Chef, SaltStack – обеспечивают высокий уровень автоматизации, однако отчеты таких систем, как правило, сводятся к бинарным выводам «соответствует/не соответствует». Это затрудняет ответ на вопросы о размере отклонения, вкладе отдельных компонентов и динамике дрейфа во времени. Русскоязычные обзоры и прикладные публикации в основном описывают функциональность и сценарии применения этих инструментов, не предлагая формализованной количественной метрики состояния конфигураций [1].

В международных работах фокус смещен к раннему обнаружению и предупреждению о дрейфе в облачных и IaC-средах, включая анализ журналов IaC/облачных сервисов и применение методов искусственного интеллекта (ИИ) и машинного обучения (МО) для распознавания расхождений (например, на данных AWS Config) – т. е. решается задача «обнаружить/сигнализировать», но не «измерить единым сопоставимым показателем» [2], [3].

Дополняя этот пласт, систематические обзоры по IaC фиксируют зрелость практик и типичные проблемы качества артефактов IaC. Так, систематический обзор исследований IaC выявляет ключевые направления (фреймворки и инструменты, эмпирические исследования, тестирование), но не предлагает унифицированной числовой метрики конфигурационного состояния на уровне узлов и ролей [4], [5]. Систематический обзор практик («do’s and don’ts») агрегирует нормы и антипаттерны из различных источников, что важно для профилактики дефектов инфраструктурного кода, однако остается в плоскости качественных рекомендаций [6]. Ряд эмпирических работ по IaC акцентирует внимание на рисках дефектов и операционных отказов, возникающих из-за некорректных изменений конфигураций [7]. В сумме эти исследования закрывают вопросы, как писать и проверять IaC, но не дают интегральной количественной метрики дрейфа для сопоставления объектов и приоритизации исправлений.

Отдельная линия – безопасность облачно-нативных систем. Современные обзоры подчеркивают, что конфигурационный дрейф является значимым фактором риска: разбалансировка параметров и политик по мере масштабирования и обновлений порождает уязвимости и несоответствие требованиям, что требует автоматизированного контроля и регулярной верификации [4]. Тем не менее, и здесь фокус – на процессах и средствах обеспечения соответствия, а не на строгой нормированной метрике дрейфа, инвариантной к типам параметров и пригодной для декомпозиции по категориям.

Итого, существует множество разных подходов к обнаружению и устранению дрейфа (в том числе с ИИ-поддержкой), а также к качеству артефактов IaC и практикам их тестирования. Однако сохраняется методический разрыв: отсутствует общепринятая числовая, интерпретируемая и сопоставимая метрика дрейфа, которая корректно учитывает частичные соответствия, поддерживает иерархическое агрегирование от проверок к категориям/ролям/кластерам и позволяет выполнять разложение вклада по областям конфигурации для принятия управленческих решений.

Цель исследования. Разработка математической модели единого показателя конфигурационного дрейфа, которая позволяет количественно оценивать степень отклонения ИТ-систем от эталонного состояния, учитывать значимость параметров и обеспечивать сопоставимость результатов на разных уровнях инфраструктуры.

Суть подхода. Предлагаемая в настоящей статье модель устраняет указанные ограничения за счет введения единого показателя дрейфа конфигурации, определяемого на отрезке [0; 1]. Эталонное состояние формируется на основе набора параметров, каждый из которых проверяется отдельным правилом. Результаты проверок интерпретируются как компоненты вектора соответствия, принимающие числовые значения от 0 до 1.

Для каждой проверки задается вес, отражающий ее значимость. Совокупность весов нормируется так, чтобы их сумма была равна единице, что обеспечивает корректность агрегации. Поддержка как бинарных правил (да/нет), так и градуированных шкал для числовых, множественных и порядковых параметров позволяет учесть частичные отклонения и сделать оценку более информативной по сравнению с традиционным бинарным подходом.

Итоговый показатель дрейфа рассчитывается как взвешенная сумма отклонений от эталона. Он может быть разложен по категориям параметров (например, безопасность, производительность, доступность), что обеспечивает интерпретируемость и позволяет выявлять ключевые источники несоответствий.

Таким образом, предлагаемое решение дополняет существующие практики системным и сопоставимым показателем состояния конфигураций. Его преимущества заключаются в возможности агрегирования результатов на разных уровнях (проверка → категория → узел/роль → кластер), интерпретируемости получаемых значений и учете частичных соответствий параметров.

Формальная постановка задач. Для описания состояния системы необходим единый числовой показатель, который:

– принимает значения на отрезке [0; 1] и однозначно определяется набором параметров конфигурации;

– формируется на основе автоматизированных проверок;

– учитывает важность параметров и допускает частичную оценку соответствия (не только «0 или 1»);

– масштабируется от одного узла до кластера и всей инфраструктуры.

Для построения такой модели необходимо:

1. Разработать свод правил, которым должны соответствовать параметры, по которым будут оцениваться конфигурации.

2. Определить категории параметров и их относительные веса.

3. Разработать шкалы оценки для различных типов параметров (бинарные, числовые, множественные, порядковые).

4. Ввести процедуру вычисления интегрального показателя соответствия и сопряженного с ним индекса дрейфа.

В дальнейшем эти элементы составляют основу модели, которая позволяет перейти от перечня отдельных проверок к единому количественному индикатору состояния конфигураций.

Правила определения параметров оценивания конфигурации. Для обеспечения корректности и достоверности вычисления показателя дрейфа необходимо формализовать требования к правилам, по которым оцениваются параметры конфигурации. Каждое правило должно удовлетворять следующим условиям:

1. Атомарность. Одно правило проверяет один параметр/факт.

2. Непересекаемость. Разные правила не описывают один и тот же факт с разных сторон.

3. Недвусмысленность. Формулировки без оценочных слов.

4. Измеримость. Результат оценки параметра приводится к числу в диапазоне [0; 1].

5. Допуски и шкала. Для градуированных правил определены пороги/интервалы и формула нормирования в [0; 1].

6. Вес и критичность. У каждого параметра есть вес.

Соблюдение этих принципов гарантирует согласованность правил и позволяет интерпретировать получаемый показатель дрейфа как корректную и сопоставимую количественную метрику.

Математическая модель. Пусть задан набор из n проверок. Для узла u каждой проверке соответствует числовая оценка соответствия $c_i(u)$ в диапазоне [0; 1]. В бинарном случае это 1 или 0. В случае правил с промежуточными значениями оценка может находиться между 0 и 1, что позволяет учитывать степень соответствия.

Каждой проверке присваиваем неотрицательный вес w_i так, чтобы их сумма равнялась единице. Тогда показатель соответствия узла $S(u) = \sum w_i c_i(u)$. Дрейф определяется как $D(u) = 1 - \sum w_i c_i(u)$. Показатель всегда находится в пределах [0; 1]: 0 – полное соответствие эталону, 1 – полное несоответствие по всем проверкам с положительными весами.

Немного расширим и усложним модель. Пусть все проверки сгруппированы по категориям (например: безопасность, производительность, доступность, соответствие стандартам). Для категории k введем вес W_k, а для каждой проверки внутри категории – локальный вес $w_{i|k}$. Тогда общий вес $w_i = W_k w_{i|k}$. Это позволит одновременно учитывать значимость категории в целом и значимость отдельных проверок внутри нее.

Пусть задано n проверок. Для узла u определим вектор соответствия:

$$\mathbf{c}(u) = \left(c_1(u), c_2(u), \ldots, c_n(u)\right)^{\mathrm{T}}, \; c_i(u) \in [0; 1].$$

Эталонному состоянию по каждой проверке соответствует значение 1, поэтому эталонный вектор

$$\mathbf{s} = 1 = (1, 1, \ldots, 1)^{\mathrm{T}}.$$

Вектор компонентных отклонений определяется как

$$\boldsymbol{\delta}_i(u) = 1 - c_i(u),$$

где $\boldsymbol{\delta}_i(u)$ – отклонение результата проверки от эталонного значения; $c_i(u)$ – результат i-й проверки для узла u в диапазоне [0; 1].

Тогда вектор отклонений имеет вид

$$\boldsymbol{\delta}(u) = \left(\delta_1(u), \delta_2(u), \ldots, \delta_n(u)\right)^{\mathrm{T}}.$$

Пусть каждой проверке назначен вес $w_i \geq 0$ так, что $\sum_{i=1}^{n} w_i = 1$ и $\mathbf{w} = (w_1, \ldots, w_n)^{\mathrm{T}}$, тогда индекс соответствия

$$S(u) = \sum_{i=1}^{n} w_i c_i(u) = \mathbf{w}^{\mathrm{T}} c(u).$$

Индекс дрейфа

$$D(u) = 1 - S(u) = \sum_{i=1}^{n} w_i \left(1 - c_i(u)\right) = \mathbf{w}^{\mathrm{T}} \delta(u).$$

Формально он эквивалентен взвешенной L_1-норме. В зависимости от требуемой чувствительности к отдельным нарушениям показатель дрейфа может определяться как взвешенная L_1- или L_2-норма вектора отклонений.

Если проверки сгруппированы по k категориям, $k = 1, 2, \ldots, K$, то удобно задать иерархические веса. Каждой категории k присваивается вес W_k, а каждой проверке внутри категории – вес $w_{i|k}$.

Для корректности агрегирования веса нормируются по уровням:

$$\sum_{i=1}^{K} W_k = 1, \; \sum_{i \in C_k} w_{i|k} = 1,$$

тогда глобальный вес проверки вычисляется как произведение весов двух уровней

$$w_i = W_k w_{i|k}.$$

При таком определении общая сумма весов всех проверок остается равной единице, т. е. нормирование сохраняется. C_k – множество проверок категории k. Тогда естественно определить соответствия категорий:

$$\bar{c}_k(u) = \sum_{i \in C_k} w_{i|k} c_i(u), \; \bar{D}_k(u) = 1 - \bar{c}_k(u).$$

И общий индекс дрейфа разложится на сумму по категориям:

$$D(u) = \sum_{k=1}^{K} W_k \bar{D}_k(u).$$

Такое представление позволяет одновременно получать агрегированное значение дрейфа и вектор значений по категориям $\left(\bar{D}_1(u), \ldots, \bar{D}_k(u)\right)$, что облегчает интерпретацию и поиск ключевых источников отклонений.

Определение типов оценки. Параметр оценки – наблюдаемый факт конфигурации (настройка, версия, состояние сервиса, множество установленных компонентов и т. п.), по которому можно вычислить числовую компоненту соответствия $c_i(u)$. В зависимости от природы параметра используются различные типы оценивания.

Выбор перечисленных типов оценки обусловлен необходимостью формализовать результаты проверок различной природы в единой числовой форме. В ИТ-инфраструктурах встречаются как параметры с однозначным состоянием (например, включен/выключен сервис), так и количественные или множественные показатели, где возможны частичные отклонения от эталона. Использование трех базовых типов оценивания – бинарного, интервального (градуированного) и наборного, позволяет охватить все основные случаи от логических условий до комплексных конфигурационных признаков. Такая классификация делает модель универсальной и обеспечивает удобство автоматизированной обработки результатов, сохраняя при этом интерпретируемость и сопоставимость оценок между различными параметрами.

1. Бинарная оценка – используется, если параметр может принимать только одно из двух состояний. Например:

– флаг/режим (*auditd включен*, *IPv4 forwarding* = 0, *запрет root-login no SSH*);

– факт наличия/отсутствия (*установлен пакет must-have*, *открыт критический порт* (*не должен быть открыт*), *включен нужный модуль ядра*);

– жесткие соответствия строк/шаблонов: (*PermitRootLogin no*).

2. Оценка с промежуточными значениями – используется в случаях, когда параметр может частично соответствовать эталону. Допускается непрерывная шкала значений от 0 до 1. Выделяются три основные разновидности:

2.1. Числовая величина: (*версия*, *задержка*, *доля свободного места*, «*дней с последнего обновления*»). Диапазон нормы: задана целевая точка r и допустимое отклонение τ.

Пусть $x = x_i(u)$ – наблюдаемое значение параметра (фактическая величина, измеренная на узле u).

Тогда оценка соответствия определяется как

$$c_i(u) = \max\left(0.1 - \frac{|x - r|}{\tau}\right).$$

Примеры:

– свободное место $\geq 20\,\%$ (штраф за каждый недостающий процент);

– загруженность ЦП.

2.2. Наборы или списки. Бинарная логика («хоть один обязательный пакет отсутствует → 0») работает для критичных позиций, но плохо отражает степень соответствия, когда список большой и часть пунктов второстепенна.

Когда следует выбирать градуированную оценку для наборов:

– список длинный (10–100 позиций) и все элементы примерно одинаково важны;

– нужны промежуточные оценки и ранжирование объектов;

– есть «обязательные», «желательные» и «запрещенные» элементы.

Разделим эталон на три подмножества:

– B_{req} – обязательные;

– B_{opt} – желательные;

– B_{ban} – запрещенные.

Пусть A – фактический набор пакетов на узле, а β_{req}, β_{opt}, β_{ban} – веса категорий; тогда оценка

$$\begin{cases} c_i = \max\left[0, \min\left(1, \beta_{\text{req}} \dfrac{|A \cap B_{\text{req}}|}{B_{\text{req}}} + \right.\right. \\ \left.\left. + \beta_{\text{opt}} \dfrac{|A \cap B_{\text{opt}}|}{B_{\text{opt}}} - \beta_{\text{ban}} \dfrac{|A \cap B_{\text{ban}}|}{B_{\text{ban}}}\right)\right]. \\ \beta_{\text{req}} + \beta_{\text{opt}} = 1. \end{cases}$$

Область применения модели. Условия, при соблюдении которых модель дает корректный результат:

1. Задан эталон.

2. Задан перечень проверок.

3. Каждая проверка удовлетворяет требованиям атомарности/непересекаемости/измеримости.

Следует подчеркнуть, что данные условия приведены в упрощенном виде для целей изложения базовой модели. Они могут быть расширены и уточнены применительно к конкретным доменам и инфраструктурам, например при формализации сложных зависимостей между параметрами или при необходимости обработки неполных данных.

Категории параметров. Для обеспечения интерпретируемости результатов проверки парамет-

ры группируются по непересекающимся и исчерпывающим категориям.

Примерный набор категорий включает:

1. SEC – безопасность доступа, уязвимости.
2. COMPL – соответствие внешним/внутренним стандартам.
3. AVAIL – доступность и отказоустойчивость (SLO/SLA).
4. PERF – производительность и емкость.
5. STACK – программный стек и целостность конфигурации (пакеты, зависимости).
6. OBS – наблюдаемость и аудит (логи, метрики, трассировка).
7. NET – сеть и периметр (файрвол, сегментация, TLS).

Категории можно адаптировать, но они должны быть непересекающимися и исчерпывающими.

Веса категорий и их относительная значимость для различных типов ролей определяются экспертными методами и могут различаться в зависимости от задач эксплуатации.[1]

Пример, демонстрирующий использование метода. В табл. 1 приведен пример распределения весов категорий для нескольких типичных ролей серверов.

Примерный список ролей:

- DB (Database) – сервер баз данных;
- WEB – веб-сервер;
- APP (Application Server) – сервер приложений;
- OBS (Open Broadcaster Software) – сервер для мониторинга.

Для демонстрации работы модели рассмотрим условный веб-сервер. Эталон задается семью категориями параметров (SEC, COMPL, AVAIL, PERF, STACK, OBS, NET). В каждой категории выбрана одна типовая проверка, отражающая ее содержание (табл. 2). Для краткости используем следующие обозначения протоколов:

- SSH – Secure Shell, протокол защищенного удаленного доступа;
- TLS – Transport Layer Security, протокол криптографической защиты соединений.

Итоговые значения:

- индекс соответствия

$$S(u) = \sum W_k c_i(u) = 0.647;$$

- индекс дрейфа

$$D(u) = 1 - S(u) = 0.353.$$

Табл. 1. Веса категорий для различных ролей
Tab. 1. Category weights for different roles

Роль	Категория						
	SEC	COMPL	AVAIL	PERF	STACK	OBS	NET
DB	0.3	0.1	0.2	0.1	0.1	0.1	0.1
WEB	0.22	0.05	0.2	0.18	0.2	0.08	0.07
APP	0.22	0.05	0.22	0.2	0.18	0.08	0.05
OBS	0.2	0.05	0.22	0.1	0.12	0.26	0.05

Наибольший вклад в дрейф вносят категории STACK (0.20) и OBS (0.08), а также частично PERF (0.048). При этом категории SEC, AVAIL и NET полностью соответствуют эталону.

Табл. 2. Расчет показателя дрейфа для веб-сервера
Tab. 2. Calculation of the drift index for a web server

Категория	Проверка	Тип оценки	c_i	W_k	$\bar{D}_k$
SEC	Root-login по SSH запрещен	Бинарная	1	0.22	0
COMPL	CIS-контролей выполнено 80 % (цель 90 %, допуск 0.2)	Интервальная	0.5	0.05	0.025
AVAIL	Readiness/Liveness-пробы настроены	Бинарная	1	0.2	0
PERF	P95-задержка 280 мс при целевом значении ≤ 200 мс и линейном штрафе до 500 мс	Интервальная	0.73	0.18	0.048
STACK	Состав пакетов (req: nginx, openssl; opt: htop; ban: telnet). Фактический состав: {nginx, htop, telnet}	Наборная	0	0.2	0.2
OBS	Аудит включен	Бинарная	0	0.08	0.08
NET	Используется TLS ≥ 1.2	Бинарная	1	0.07	0

Таким образом, модель позволяет не только получить агрегированную оценку, но и выделить категории, которые ответственны за наибольшие отклонения. Это делает показатель дрейфа интерпретиру-

[1] В данной статье этот аспект рассматривается лишь в ознакомительном ключе. Детальная методика калибровки весов и примеры их практического использования будут представлены в последующих работах.

емым и полезным для выявления приоритетных направлений корректировки конфигурации.

Заключение. Предложенная математическая модель позволяет перейти от фрагментарного перечисления нарушений к единому показателю дрейфа конфигурации, применимому на всех уровнях ИТ-инфраструктуры – от отдельных узлов до кластера в целом. В отличие от существующих решений, ограничивающихся бинарной логикой и разрозненными проверками, модель поддерживает градуированные оценки, учитывает значимость параметров и обеспечивает декомпозицию по категориям.

Формальная база, основанная на свойствах ограниченности, аддитивности и устойчивости к изменениям шкал, делает предложенный показатель универсальным инструментом для анализа состояния систем. Он может использоваться как для оперативной оценки текущего уровня дрейфа, так и для стратегического сравнения ролей и узлов, выявления зон риска и планирования работ.

Перспективы развития связаны с дальнейшей калибровкой весов и порогов на основе экспертных процедур и нормативных документов, а также с интеграцией модели в процессы мониторинга и практики системного администрирования. Это позволит формировать систему раннего предупреждения о нарастающем дрейфе, оптимизировать распределение ресурсов на устранение несоответствий и повысить предсказуемость эксплуатации.

Таким образом, предложенное решение формирует строгий теоретический фундамент и в то же время имеет высокую прикладную ценность, дополняя существующие инженерные практики управляемым и дающим сопоставимые результаты показателем качества конфигураций.

Список литературы

1. Костромин Р. О. Сравнительный обзор средств управления конфигурациями ресурсов вычислительной среды функционирования цифровых двойников // Информационные и математические технологии в науке и управлении. 2021. № 1(21). С. 131–145. doi: 10.38028/ESI.2021.21.1.011.

2. Thiyagarajan G., Bist V., Nayak P. AI-Driven configuration drift detection in cloud environments // Intern. J. of Communication Networks and Inform. Security (IJCNIS). 2024. Vol. 16, no. 5. P. 721–743. URL: https://www.ijcnis.org/index.php/ijcnis/article/view/7898 (дата обращения: 24.09.2025).

3. Pohjola O. Mitigating configuration drift in infrastructure-as-code systems // Master's Thesis, Aalto University. 2025. URL: https://aaltodoc.aalto.fi/bitstreams/6a2f20a2-d7cb-4037-9453-22d28fb0f10a/download (дата обращения: 24.09.2025).

4. Security in cloud-native services: A survey / T. Theodoropoulos, L. Rozsa, C. Benzaid, P. Gray, E. Marin, A. Makris, L. Cordeiro, F. Diego, P. Sorokin, M. Di Girolamo, P. Barone, T. Taleb, C. Cérin // J. of Cybersecurity and Privacy. 2023. Vol. 4, no. 3. P. 758–793. doi: 10.3390/jcp4030034.

5. Rahman A., Mahdavi-Hezaveh R., Williams L. A systematic mapping study of infrastructure as code research // Inform. and Software Technol. 2019. Vol. 108. P. 65–77. doi: 10.1016/j.infsof.2018.12.004.

6. The do's and don'ts of infrastructure code: A systematic gray literature review / I. Kumara, M. Garriga, A. U. Romeu, D. Di Nucci, D. A. Tamburri, W.-J. van den Heuvel, F. Palomba // Inform. and Software Technol. 2021. Vol. 137, no. 9. P. 106593. doi: 10.1016/j.infsof.2021.106593.

7. Rahman A., Farhana E., Williams L. The 'as code' activities: development anti-patterns for infrastructure as code // Empirical Software Engin. 2020. Vol. 25, no. 5. P. 3430–3467. doi: 10.1007/s10664-020-09841-8.

Информация об авторах

Моисеев Кирилл Олегович – магистрант кафедры информационных систем СПбГЭТУ «ЛЭТИ». E-mail: vihuhool@vk.com

Трофимова Анастасия Олеговна – магистрант кафедры информационных систем СПбГЭТУ «ЛЭТИ». E-mail: anastasiia_trofimova02@mail.ru

References

1. Kostromin R. O. Sravnitel'nyj obzor sredstv upravlenija konfiguracijami resursov vychislitel'noj sredy funkcionirovanija cifrovyh dvojnikov // Informacionnye i matematicheskie tehnologii v nauke i upravlenii. 2021. No. 1(21). S. 131–145. doi: 10.38028/ESI.2021.21.1.011. (In Russ.).

2. Thiyagarajan G., Bist V., Nayak P. AI-Driven configuration drift detection in cloud environments // Intern. J. of Communication Networks and Inform. Security (IJCNIS). 2024. Vol. 16, no. 5. P. 721–743. URL: https://www.ijcnis.org/index.php/ijcnis/article/view/7898 (data obrashhenija: 24.09.2025).

3. Pohjola O. Mitigating configuration drift in infrastructure-as-code systems // Master's Thesis, Aalto University. 2025. URL: https://aaltodoc.aalto.fi/bitstreams/6a2f20a2-d7cb-4037-9453-22d28fb0f10a/download (data obrashhenija: 24.09.2025).

4. Security in cloud-native services: A survey / T. Theodoropoulos, L. Rozsa, C. Benzaid, P. Gray, E. Marin, A. Makris, L. Cordeiro, F. Diego, P. Sorokin, M. Di Girolamo, P. Barone, T. Taleb, C. Cérin // J. of Cybersecurity and Privacy. 2023. Vol. 4, no. 3. P. 758–793. doi: 10.3390/jcp4030034.

5. Rahman A., Mahdavi-Hezaveh R., Williams L. A systematic mapping study of infrastructure as code research // Inform. and Software Technol. 2019. Vol. 108. P. 65–77. doi: 10.1016/j.infsof.2018.12.004.

6. The do's and don'ts of infrastructure code: A systematic gray literature review / I. Kumara, M. Garriga, A. U. Romeu, D. Di Nucci, D. A. Tamburri, W.-J. van den Heuvel, F. Palomba // Inform. and Software Technol. 2021. Vol. 137, no. 9. P. 106593. doi: 10.1016/j.infsof.2021.106593.

7. Rahman A., Farhana E., Williams L. The 'as code' activities: development anti-patterns for infrastructure as code // Empirical Software Engin. 2020. Vol. 25, no. 5. P. 3430–3467. doi: 10.1007/s10664-020-09841-8.

Information about the authors

Kirill O. Moiseev – Master's degree student, Department of Information Systems, Saint Petersburg Electrotechnical University.
E-mail: vihuhool@vk.com

Anastasiia O. Trofimova – Master's degree student, Department of Information Systems, Saint Petersburg Electrotechnical University.
E-mail: anastasiia_trofimova02@mail.ru

Статья поступила в редакцию 29.10.2025; принята к публикации после рецензирования 04.12.2025; опубликована онлайн 29.01.2026.

Submitted 29.10.2025; accepted 04.12.2025; published online 29.01.2026.