https://doi.org/10.31649/1997-9266-2025-181-4-108-117

УДК 004.4

Ю. В. Сторожук[1]
О. О. Коваленко[1]

МЕТОД ВИЗНАЧЕННЯ ЦІЛЬОВИХ ПОКАЗНИКІВ ПРОДУКТИВНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

[1]Вінницький національний технічний університет

Розробка методів визначення цільових показників для вимірювання рівня якості, надійності продуктивності програмного забезпечення є одним з актуальних напрямів в галузі інформаційних технологій та, зокрема, в програмній інженерії. Серед відомих методів можна виділити метод збалансованих показників управління, SMART-методики тощо. Методологія GIST (Goals, Ideas, Step-Projects, Tasks) – це метод продуктового планування, розроблений І. Гіладом, колишнім продакт-менеджером Google. Вона спрямована на зменшення накладних витрат на управління, підвищення швидкості розробки та створення продуктів, які краще відповідають потребам ринку та їхній предметній області. Визначення цільових показників продуктивності є актуальним питанням для сучасних DevOps-команд. Вони часто опиняються між двома крайнощами — надмірним збиранням тисяч часових рядів, що збільшує витрати та шум в інформаційних повідомленнях щодо виявлення зменшення продуктивності, і навпаки — браком даних для своєчасного виявлення регресії продуктивності. Наявні «легкі» чек-листи (Four Golden Signals, RED/USE тощо) спрощують старт, але залишаються статичними й не враховують бізнес-цілі, критичність сервісу та стадію життєвого циклу. У статті запропоновано удосконалити метод GIST (Goal Impact Stage Template) для вимірювання продуктивності програмного забезпечення (GISTSP). Мета досліджень — створити й оцінити метод формування цільових показників на основі методології вибору метрик GIST. Такий метод здатний за кілька кроків генерувати мінімальний, але достатній і обґрунтований набір показників для конкретного сервісу. Запропонований метод ґрунтується на 4-рядному «паспорті сервісу» та бібліотеці шаблонів Core + Plus-метрик. Використовуючи правило «додай тільки те, що справді потрібно», GISTSP автоматично генерує конфігурацію Prometheus/Grafana й алерт-правила. Експериментальне порівняння з відомими методами Four GS, RED/USE та AWS W-A виконані на чотирьох стендах (Web-API, Queue, Stream, Batch). Результати показали, що покриття необхідних показників зросло з 78 % до 92 %; добовий обсяг даних скоротився на 60 %; час налаштування зменшився у 4 рази, при цьому міра виявлення регресії не погіршилася. Результати дослідження свідчать про те, що GISTSP забезпечує баланс між простотою старту та гнучкістю, необхідною для різних бізнес-цілей і середовищ.

Ключові слова: програмна інженерія, тестування програмного забезпечення, тестування продуктивності програмного забезпечення, показники продуктивності програмного забезпечення, метод цільових показників продуктивності програмного забезпечення, DevOps, алгоритми, шаблони, бази знань для тестування продуктивності, IT-проєкт.

Вступ

Розробка методів визначення цільових показників для вимірювання рівня якості, надійності продуктивності програмного забезпечення є одним з актуальних напрямів в галузі інформаційних технологій та зокрема в програмній інженерії [1]. Серед відомих методів можна виділити такі як метод збалансованих показників управління, SMART-методики, GIST тощо [2], [3].

Методологія GIST (Goals, Ideas, Step-Projects, Tasks) — це метод продуктового планування, розроблений І. Гіладом, колишнім продакт-менеджером Google [4]. Вона спрямована на зменшення накладних витрат на управління, підвищення швидкості розробки та створення продуктів, які краще відповідають потребам ринку та їхній предметній області.

У більшості DevOps-команд сьогодні є або «метричний голод» (бракує даних для прийняття рішень), або «метричний надлишок» (тисячі часових рядів, за якими важко аналізувати). Популярні чек-листи — Four Golden Signals, RED/USE, Datadog Core Metrics — істотно спрощують старт,

але залишаються орієнтованими лише на одну-дві цілі (здебільшого надійність); є статичними — мають однаковий набір метрик для різних стадій створення, інтеграції та впровадження програмних продуктів; а також є нечутливими до бізнес-цілей та ризиків сервісів.

У результаті команди або збирають «усе, що є», підтримуючи високу плату за сховище та «шум» в різноманітних інформаційних повідомленнях та візуалізації, або, навпаки, — пропускають регресії, тому що «потрібна» метрика не була додана вчасно.

Мета роботи — створити й оцінити метод формування цільових показників на основі методології вибору метрик GIST (Goal Impact Stage Template), здатний за кілька кроків генерувати мінімальний, але достатній і обґрунтований набір показників для конкретного сервісу.

Для досягнення мети необхідно проаналізувати набір атрибутів, сформувати правила активації та бібліотеку шаблонів, реалізувати необхідну спеціалізовану програму для формування та візуалізації цільових показників. Порівняння запропонованого методу з базовими дасть можливість оцінити ефективність нової методики формування показників оцінювання продуктивності програмного забезпечення.

Результати дослідження

Аналіз основних складових методології GIST для продуктивності програмного забезпечення виконаємо за допомогою деталізації сутності кожної складової з фокусуванням уваги на реалізацію визначеної складової для вимірювання продуктивності програмного забезпечення.

Goals (Цілі) — це довгострокові стратегічні цілі, які компанія прагне досягти взагалі та виділяє ті, які можна отримати за допомогою використання та удосконалення програмного забезпечення. Вони визначають напрямок, мотивацію та спосіб вимірювання прогресу щодо отримання цільових результатів. Цілі мають бути конкретними, вимірними, досяжними, релевантними та обмеженими в часі (SMART-критерії). Часто для визначення цілей використовують фреймворк OKR (Objectives and Key Results). Для аналізу продуктивності повинні бути визначені основні пріоритетні напрями та показники предметної області.

Ideas (Ідеї) — це гіпотетичні способи досягнення поставлених цілей. Ідеї можуть надходити з різних джерел (зворотний зв’язок від клієнтів, аналіз ринку, внутрішні мозкові штурми). Важливо, щоб ідеї пов’язувалися з конкретними цілями компанії та відображалися в реалізації програмного продукту. GIST передбачає постійний потік ідей та їхню пріоритезацію на основі оцінки впливу (Impact), впевненості (Confidence) та зусиль (Effort, ICE-scoring або подібних методів). Для оцінювання продуктивності програмного забезпечення необхідно сформувати показники, пов’язані як з предметною областю програмного продукту, так і з технологіями його реалізації.

Step-Projects (Крок-проєкти) — це міні-проєкти, які реалізують частину ідеї, дозволяючи її перевірити та валідувати. Замість того, щоб одразу будувати повноцінну функцію, GIST пропонує розбивати великі ідеї на менші, керовані «кроки», що дозволяють швидко отримати зворотний зв’язок і зменшити ризики. Це схоже на концепцію мінімально життєздатного продукту (MVP). Для наших досліджень важливо сформувати дашборди, що містять збалансовані цільові показники, які отримуватимуться за достатньо короткий період з використанням сучасних програмних технологій.

Tasks (Завдання) — це щоденні, конкретні дії, які реалізують крок-проєкт. Завдання призначаються членам команди, їх прогрес відстежується. Саме тому запропонований метод на основі методології GIST повинен підвищити ефективність роботи команди зі створення якісного програмного забезпечення.

Розглянемо відомі методи формування наборів показників продуктивності, які представлені в зведених промислових інструкціях щодо вимірювання продуктивності.

Four Golden Signals (Google SRE) [5] — Latency, Traffic, Errors, Saturation — найпопулярніший набір для HTTP-сервісів. Перевага такого сервісу — простота формування метрик. Обмеження сервісу — не охоплює асинхронні процеси й не враховує бізнес-цілі.

RED/USE [6] — (Rate, Errors, Duration) фокусується на запитах рівня API; USE (Utilisation, Saturation, Errors) — на ресурсах (CPU, диск, мережа). Для формування показників пропонується чеклист «по три метрики» на кожний об’єкт спостереження, проте ігнорується стадія життєвого циклу та витрати енергоспоживання.

Datadog «Standard Metrics» (Web-API) [7] — пропонується список з 10 «базових» request-метрик (latency-p95, 4xx, 5xx, RPS тощо). Шаблон зручний для швидкого старту, але не масштабований на стримінгові та асинхронні процеси.

AWS Well-Architected “Key Workload Metrics” [8] — використовується для Web, Queue, Batch робочих навантажень, надаються окремі таблиці core-метрик. Модель статична, не враховує різні цілі предметних областей та критичні процеси сервісу.

Використання шаблонів у DevOps-інструментах дозволяє фахівцям структурувати свою діяльність, але має свої недоліки. Так, шаблони Prometheus Best Practices описують «http_server_ requests_seconds», «process_cpu_seconds_total» тощо як «має бути», але не дають правил, коли та як деталізувати та відбирати показники.

Шаблони New Relic Service Blueprints — надають готові дашборди для Web, Worker, Mobile, але не враховують зв’язок з цілями та пріоритетністю. Набір метрик фіксований; цілі та вплив на процеси не враховуються.

OpenTelemetry Semantic Conventions v1.22 надають єдині імена для HTTP, працюють з повідомленнями та асинхронними процесами, що спрощує інструментарій, проте вибір конкретних груп показників залишається на розсуд команди.

Виконавши аналіз відомих рішень для формування показників продуктивності можна зробити висновок, що більшість шаблонів орієнтовані лише на надійність, не охоплюючи цілі предметної області — такі, наприклад, як «контроль витрат» або «зелена енергетика». Різноманітним сервісам часто призначають однаковий набір метрик, що збільшує шум і витрати і не враховує пріоритетність сервісів. Комплекти показників не змінюються між різними стадіями створення, інтеграції та запровадження проєктів, хоча потреби у фахівців команди на цих етапах різні. Жорсткі шаблони важко комбінувати, наприклад, Web-API працює з внутрішньою чергою; а додавання GPU-Inference не має ручного редагування, що призводить до відхилення рішення щодо їх використання.

Зазначені недоліки підтверджують необхідність створення методу цільових показників, який зберігає простоту «core + plus», але додає виміри методології GIST.

В табл. 1 подано порівняння популярних підходів та методу вибору цільових показників на основі GIST – GISTSP – GIST Software Performance.

Таблиця 1

Порівняння підходів щодо формування показників вимірювання продуктивності

Підхід / Наявність інструментів	Goal-орієнт	Impact	Stage	Бібліотека шаблонів |
Four Golden Signals	—	—	—	Web-API
RED / USE	—	—	—	Ресурси /HTTP
Datadog Std Metrics	частково	—	—	Web-API
AWS W-A Metrics	частково	—	—	Web / Queue / Batch
GISTSP	+	+	+	4 цільових напрями з можливістю розширення

Аналіз табл. 1 дозволяє підсумувати щодо обґрунтованості використання GISTSP, який поєднує переваги шаблонних методів (низький поріг входження) з гнучкістю, необхідною для різних бізнес-цілей, стадій та критичності сервісу. Метод формування цільових показників базується на двох етапах — формування паспорта реалізації методу як сервісу з чотирьох атрибутів (Goal, Impact, Stage, Template) за контекстом і таблиці відповідності «паспорт → набір метрик», яка автоматично генерує конфігурацію моніторингу.

Паспорт оформлюється у YAML-файлі service.yaml.

```
passport:
goal  : regression   # regression | reliability | cost | greenIT
impact : high      # high | medium | low
stage : prod      # dev | staging | prod
template: web-api    # web-api | async-queue | stream | batch.
```

де Goal — домінуюча операційна або бізнес-ціль, заради якої збираються метрики;
Impact — бізнес-критичність сервісу. Визначає, скільки додаткових (“plus”) метрик дозволено;
Stage — фаза життєвого циклу, що впливає на частоту збору та глибину агрегування;
Template — архітектурний/функціональний тип сервісу, що задає базовий (core) набір метрик.

Шаблони метрик можуть бути представлені таким чином [8], [9]:

```
Template: web-api
  Core : latency-p95, error-rate, RPS
  Plus : latency-p99, backlog-depth, $/RPS, power/W
```

Template: async-queue
Core : queue-size, dequeue-rate, error-rate
Plus : consumer-lag, CPU-util, $/msg

Template: stream
Core : e2e-lag, records-in/out, error-rate
Plus : watermarks, partition-skew, energy/J/msg

Template: batch
Core : job-duration, throughput, success-ratio
Plus : $/job, energy/J/job, peak-mem

Правила активації Plus-метрик можна представити як R функції (приклад подано для фінансового моніторингу та «зеленою економікою»).

R1 Якщо impact = high →увімкнути всі Plus метрики.
R2 Якщо stage = dev →вимкнути високі квантілі (latency-p99, e2e-lag) і енергетичні показники.
R3 Якщо goal = cost → додати вартісні метрики ($/RPS, $/job). (1)
R4 Якщо goal = greenIT → додати енергетичні метрики (energy/J/*, power/W).
R5 Якщо одночасно stage = prod та impact ≠ low → збільшити частоту збору core-метрик удвічі

де Impact — рівень пріоритетності; Stage — визначення етапу для оцінювання продуктивності; Goal — цільові показники предметної області.

Для реалізації запропонованого методу використовуються інструменти для вибору необхідних метрик, генерування показників та представлення їх на панелях дашбордів, виконання CI/CD інтеграції, формування бібліотеки шаблонів та власних показників.

Отже, GISTSP забезпечує прозорий процес вибору метрик «у два кліки»; мінімальний стартовий набір, що відповідає цілям; низький поріг входження завдяки готовим шаблонам.

Запропонований метод визначення цільових показників GISTSP задумано як «10-хвилинний» спосіб перейти від нульової інструментації до осмисленого переліку метрик. Керівним принципом є поділ на Core-метрики — універсальний мінімум, який дає до 80 % користі; Plus-метрики — додаються лише якщо це виправдано цілями, критичністю навантаження або використанням чи середовищем.

Розглянемо приклад оцінювання продуктивності для сервісу checkout-api (критичний, продакшен, ціль — надійність).

Паспорт методу можна представити як:

passport:
goal: reliability
impact : high
stage : prod
template: web-api

Алгоритм дій подамо такими етапами:

1. Core(web-api) = {latency-p95, error-rate, RPS}.
2. Impact = high → додаємо Plus = {latency-p99, backlog-depth, $/RPS, power-W}.
3. Stage = prod → scrape-інтервал 5 с.

Запровадження запропонованого методу дозволяє отримати 5 метрик замість типових 25…30; економія дискового обсягу ≈ 80 %, час налаштування — 8 хв.

Переваги методу полягають в простоті, адаптивності, зменшенні часу для моніторингу бізнес-цілей та інтеграції. Зокрема, такі переваги можна деталізовувати за такими параметрами:

– простота — лише 4 поля YAML;
– адаптивність — до цілей / критичності/пріоритетності;

– зменшення часу моніторингу загальної вартості продукту, вартості процесів вимірювання показників продуктивності, інтеграції та відповідності до бізнес-цілей, що є цінним для власників продукту.

Процеси інтеграції оцінюються через один скрипт.

Якщо набір шаблонів невеликий, то необхідно розробляти нові шаблони, а також правила і показники для предметних областей.

Для підтвердження науково-практичних гіпотез проведено експерименти. В табл. 2 подані основні гіпотези формування методу цільових показників продуктивності ПЗ.

Запропоновані гіпотези можуть бути перевірені під час експериментів збору даних, здійснення моніторингу та формування ключових показників.

Експеримент виконано з такими обмеженнями валідності як:

– GIST-утиліта й базові шаблони формують конфігурацію в Prometheus-форматі, тому що різна семантика назв могла б вплинути на збір точок. Для мінімізації всі метрики проходили єдину валідацію «scrape-lint».

– налаштування здійснювалось у різних порядках (Latin-square), щоб уникнути ефекту «другого разу швидше».

– усі регресії (CPU-throttle, latency-delay) задавалися однаковими Helm-chartами; випадкове псевдовведення даних контролюється спеціальним плагіном.

Таблиця 2

Гіпотези та показники оцінювання

Код гіпотези	Формулювання	Ключовий показник
H1 (Coverage) Оцінювання покриття рівнів обслуговування	GISTSP забезпечує ≥ 90 % покриття потрібних SLI для заданої цілі Goal і не поступається жодному з базових підходів	Coverage = (SLI вибрано / SLI потрібно) ·100 %
H2 (Volume) Рівень обсягу зібраних даних	GISSPT скорочує обсяг зібраних даних ≥ 50 % порівняно з кращим з базових чек-листів	ΔVolume = (Samplesbaseline – SamplesGIST)/Samplesbaseline
H3 (TTA) Час налаштування моніторингу	Час налаштування моніторингу (time-to-adopt, TTA) у GISTSP ≤ 30 % від ручного вибору метрик	TTA — хвилини від «порожньо» до «дані в Grafana»
H4 (Detect) Виявлення регресії	З тим самим рівнем покриття GISTSP не погіршує F1-міру виявлення регресій більше ніж на 5 %	F1 (Precision/Recall аномалій)

В табл. 3 подано дані стендів та робочих навантажень для експерименту. Обмеження конструктивної валідності полягали в тому, що в експерименті розраховувалися тільки показники, визначені командою для бізнес-цілей. Але існує ризик, що деякі команди трактуватимуть потрібний набір інакше.

Таблиця 3

Стенди, шаблони, технології та робочі навантаження

Код стенду	Шаблон GISTSP	Технологія	Навантажувач / патерн
S1-Web	web-api	3-подовий Go-сервіс у K8s	k6, 200→1000 RPS, bursty
S2-Queue	async-queue	RabbitMQ 3.13, 2×consumer	rq-perf, steady 10k msg/s
S3-Stream	stream	Kafka 3.6 + Flink 1.17	fakestream, 5 MB/s, diurnal
S4-Batch	batch	Spark-standalone ETL job	Airflow, job/15 хв

Порівняння здійснювалось за кількістю записів, а не фактичний розмір у байтах; у різних TS-базах стиснення відрізняється. Проте коефіцієнти стиснення дозволяють оцінити пропорційні обсяги записів і вважати отримати результати достовірними. Для оцінювання типу навантажень використано чотири шаблони.

Експерименти виконано в сервісі Amazon Kubernetes; Інші сервіси можуть мати інші затрати на зберігання.

Для кожного стенду формується YAML-паспорт (Goal, Impact, Stage, Template).

Табл. 4 демонструє очікувані пороги прийняття результатів підтвердження гіпотез.

Таблиця 4

Очікувані пороги прийняття результатів підтвердження гіпотез запропонованого методу

Гіпотеза	Що доводимо	Приймаємо, якщо виконано всі умови
H1 — Coverage «GIST покриває потрібні показники не гірше, ніж найкращий з базових списків»	1) Середнє покриття GISTSP ≥ 90 %. 2) GISTSP не гірше найкращого базового підходу	Mean(CoverageGIST) ≥ 0,90 ΔCoverage = CoverageGIST − CoverageBest ≥ 0; p < 0,05 (односторонній парний t-тест)
H2 — Volume «GISTSP істотно скорочує обсяг даних»	Обсяг зібраних точок у GISTSP не перевищує половини обсягу найкращого базового підходу	VolumeGIST ≤ 0,5 × VolumeBest p < 0,05 (односторонній парний t-тест або Wilcoxon)
H3 — Time-to-Adopt (TTA) «Налаштувати GISTSP значно швидше, ніж вручну»	Середній час налаштування GISTSP становить максимум 30 % від часу «ручного» процесу	TTAGIST / TTAmanual ≤ 0,30 p < 0,05
H4 — Detection quality «GISTSP не погіршує якість виявлення регресій»	Різниця у F1-мірах між GISTSP і найкращим базовим підходом не перевищує ±0,05	ΔF1 = 0,05

Приклад вибору цілей S_1—S_4 представлено такою формулою:

$$\begin{aligned} &S1 \text{ — reliability, high, prod;} \\ &S2 \text{ — regression, medium, staging;} \\ &S3 \text{ — cost, medium, prod;} \\ &S4 \text{ — greenIT, low, dev.} \end{aligned} \quad (2)$$

Загальний результат експерименту повинен підтвердити або не підтвердити головну мету запровадження методу формування цільових показників оцінювання продуктивності програмного забезпечення «мінімум метрик — максимум користі» без втрати здатності виявляти проблеми. Використано 4 стенди, 20 релізів, 80 спостережень.

Очікувані результати можна трактувати таким чином:

– 90 % покриття (H1) — мінімум, який є «достатнім» для контролю показників продуктивності програмного забезпечення;

– 50 % зменшення обсягу показників(H2) робить економію сховища й трафіку помітною навіть для невеликих кластерів;

– 30 % від часу ручного налаштування (H3) означає, що налаштування GISTSP приблизно у 3 рази швидше;

– ΔF1 = 0,05 (H4) — типова межа «не гірше», яку застосовують у практичних дослідженнях системної програмної та комп'ютерної інженерії.

Алгоритм виконання експерименту можна представити 4-ма етапами.

1. Формування вікі-карти, завдання показників таймера; фіксується час та показники до появи дашборду з усіма core-метриками.

2. Робоча фаза — перевіряються 20 «релізів»: 15 нормативних, 5 з навмисними регресіями (штучно вставлений 30 % CPU-тротлінг або затримка +50 %).

3. Збір показників здійснюється за виконанням таких стадій:

– Samples — визначення кількості точок/хв/под, розрахунок добового обсягу у байтах;

– Coverage — перевірка переліком «потрібно для Goal»;

– F1 — реалізація алгоритму виявлення точок змін на часових рядах (RuLSIF).

4. Перемикання підходу → повтор кроків 1—3; дизайн «within-subject».

Експеримент доцільно виконати з командою інженерів різного рівня досвіду аби нівелювати різницю у навичках інженерів.

Для оцінювання результатів експерименту використано такі методи обробки та статистики.

Для H1—H3 — парний t-тест (n =20 релізів) + ефект Кліффа δ.

Для H4 — Wilcoxon signed-rank (непараметрично).

Контроль багатьох гіпотез — Holm-Bonferroni.

Розміри ефектів визначаються за такою шкалою:

0,2 — низький; 0,5 — середній; ≥ 0,8 — високий.

Усереднення метрик «Samples» та «Coverage» виконується окремо для кожного стенду, далі проводиться загальна оцінка (random-effects model), тому що сервіси різнорідні.

Детальніше результати подані в таблицях результатів за кожною гіпотезою. Так в табл. 5 подано рівень покриття бізнес-цілей.

Таблиця 5

Покриття бізнес-цілей (H1)

Стенд	Goal	Coverage, % GISTSP	Coverage, % Best baseline*
S1-Web	reliability	93	90 (Four GS)
S2-Queue	regression	92	88 (RED/USE)
S3-Stream	cost	91	70 (AWS W-A)
S4-Batch	greenIT	92	64 (AWS W-A)
Середнє	- experts@naqa.gov.ua	92	78

Всі результати за покриттям за використання методу з методологією GISTSP показали більше 90 %, порівнювальні методи — тільки 90 % (Four GS) для першого стенду і менше для всіх інших.

В табл. 6 подано результати експерименту за оцінюванням обсягу зібраних даних.

Результати середнього скорочення в порівнянні з іншими методами для методології GISTSP складає −60 %** ($p < 0{,}001$). Для кожного з чотирьох експериментальних стендів (S1 — Web, S2 — Queue, S3 — Stream і S4 — Batch) визначено добовий обсяг метрик за трьома базовими підходами — Four Golden Signals, RED/USE та AWS Well-Architected. З цих трьох значень, вибрано найменше; воно відмічене як «Baseline» для цього стенда.

Таблиця 6

Обсяг зібраних даних (H2)

Стенд	Samples/добу на pod	Four GS	RED/USE	AWS W-A	GISTSP
S1	(Web)	14 400	21 600	16 200	5 760
S2	(Queue)	12 960	19 440	18 000	6 480
S3	(Stream)	18 000	23 760	20 160	7 200
S4	(Batch)	8 640	14 400	12 960	4 320

Далі розраховано відносну економію GISTSP за формулою

$$\Delta\text{Volume} = (\text{Volume\_Baseline} — \text{Volume\_GISTSP}) / \text{Volume\_Baseline}\ , \qquad (3)$$

де ΔVolume — відносна економія GISTSP; Volume_Baseline — найменше значення добового обсягу метрик на стенді; Volume_GISTSP — найменше значення добового обсягу метрик на стенді GISTSP.

Середнє значення — 0,60, тобто 60 % скорочення витрат. Статистичну значущість різниці між парами «Baseline vs GISTSP» перевірили парним t-тестом ($n = 4$), отримавши $p < 0{,}001$. Така вірогідність p свідчить про найкращий рівень скорочення витрат за запропонованим методом.

В табл. 7 подано результати використання методу командою інженерів для вимірювання продуктивності за шаблонами.

Таблиця 7

Оцінювання часу роботи інженерів за різними методами

Учасник (N = 5)	TTA, хв (Four GS)	TTA, хв (ручний підхід**)	TTA, хв (GISTSP)
U1	22	48	12
U2	24	45	10
U3	21	42	13
U4	23	47	11
U5	25	52	12
Середнє	23	47	11.6

Примітки: **— «ручний», тобто інженер сам підбирає метрики поза шаблонами; а також вибрані методи за кращими показниками покриття; абревіатура TTA — часовий інтервал формування метрик; U*i*…*n* — учасник.

Розрахунок абсолютної різниці за якістю регресії між найкращими показниками інших методів і показниками запропонованого методу виконувався за формулою

$$\Delta F_1 \text{ (GISTSP – Best)} = -0{,}02 \ (95\ \%\ \text{CI} - 0{,}04;\ 0{,}01) \Rightarrow \text{у межах } \pm 0{,}05. \quad (4)$$

Якість виявлення регресії подана в табл. 8.

Таблиця 8

Якість виявлення регресій (H4)

Стенд	F_1 (Four GS)	F_1 (RED/USE)	F_1 (AWS W-A)	F_1 (GISTSP)
S1	0,86	0,88	0,84	0,87
S2	0,80	0,82	0,78	0,81
S3	0,83	0,81	0,79	0,82
S4	0,76	0,75	0,77	0,75

Результати стандартного непараметричного тесту за 80 показниками різниці свідчать про те, що за показником ймовірності запропонований метод GISTSP справді відстає від інших методів на 0,05 або більше, до 1,8 %. Оскільки ця ймовірність менша за загальноприйнятий поріг 5 %, ми можемо відкинути нульову гіпотезу і зробити висновок, що GISTSP не поступається базовим підходам за точністю виявлення регресій.

Отже, можна зробити такі висновки:

1. GISTSP досягає рівня покриття, еквівалентного або вищого за найкращі з наявних чек-листів, що зафіксовано за 40 % набором метрик.

2. Найбільший виграш запропонований метод має у «нетипових» цілях предметних областей (перевірено на метриках фінансового аналізу та «зеленої економіки»).

3. Суттєве скорочення обсягів (–60 %) прямо конвертується у зниження витрат на сховище Prometheus/Thanos ~ 55 USD/100 запитів на·місяць.

4. Час налаштування — ключовий аргумент для невеликих команд: 10…13 хв. проти 45…50 хв. ручного налаштування конфігурації.

Результати створення та запровадження методу цільових показників продуктивності програмного забезпечення свідчать про те, що командам із переважно Web-трафіком достатньо Core-набору GISTSP, Plus-метрики (latency-p99, backlog) необхідно використовувати для high-impact сервісів. Команди також мають можливість працювати з базовими шаблонами, мати швидкий зворотний зв’язок.

Висновки

GISTSP дозволяє сформувати систему показників, яка збалансовує загальні підходи з невеликим набором метрик та буде альтернативою для методів з онтологічними громіздкими підходами. Такий метод пропонує практичний баланс за рахунок показників як високе покриття фактичних показників, мінімальний обсяг даних і швидке впровадження. Відкрита бібліотека шаблонів та утиліта gist-apply дозволяють реалізувати метод для впровадження, використання й подальшого розвитку спільнотою.

За результатами досліджень розроблено метод формування цільових показників продуктивності GISTSP, що представлений як 4-рядковий «паспорт сервісу» й бібліотека шаблонів Core + Plus.

У порівнянні з найближчими «легкими» підходами (Four Golden Signals, RED/USE, AWS W-A) запропонований метод дозволить отримати такі показники ефективності:

– покриття потрібних фактичних показників зросло з 78 % до 92 %;

– обсяг зібраних даних скоротився на 60 %;

– час налаштування скоротився у 4 рази (≈ 12 хв. проти ≈ 47 хв. вручну);

– якість виявлення регресій (F_1) залишилася статистично не гіршою ($\Delta \leq 0{,}02$).

Результати експерименту дозволяють стверджувати, що атрибути Goal, Impact, Stage та Template виявилися достатніми.

Мінімальний набір метрик, сформований GISTSP, утримує якість виявлення регресій на рівні 0,80…0,87, що відповідає базовим показникам.

Середня економія сховища (60 %) перевищує поріг «цінність / вартість» для більшості команд (≈ 50 GB на 100 запитів на·місяць).

Метод успішно працює для Web-API, Queue, Stream та Batch.

Запровадження запропонованого методу дозволяє командам інтегрувати GISTSP у CI/CD як окремий сервіс планування показників продуктивності та зменшити загальну вартість володіння програмним продуктом та процесів її моніторингу.

Власник продукту отримує швидкий спосіб відобразити бізнес-пріоритети (Impact, Goal) у технічних показниках.

Метод формування цільових показників дозволяє працювати з інструментом Plus-метрики без переходу на громіздкі фреймворки.

Плани подальших досліджень передбачають розширення бібліотеки шаблонів, формування системи автоматичного перемикання метрик за динамічними подіями, інтеграцію дашбордів для підвищення рівня візуалізації, дослідження GISTSP для програмного забезпечення різних предметних областей.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] V. S. Yakovyna, M. M Seniv, I. I. Symets, and N. B. Sambir, “Algorithms and software suite for reliability assessment of complex technical systems,” *Radio Electronics, Computer Science, Control*, no. (4), pp. 163-177, 2020. https://doi.org/10.15588/1607-3274-2020-4-16 .

[2] T. Murphy, and K. Cormican, “Towards holistic goal centered performance management in software development: lessons from a best practice analysis,” *IJISPM*, vol. 3, no. 4, pp. 23-36, Feb. 2022.

[3] ISO/IEC, “ISO/IEC 25010:201, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models,” 2011. [Electronic resource]. Available: https://www.iso.org/standard/35733.html#lifecycle .

[4] I. Gilad the GIST Board and Other GIST Tools. Tech. Rep., Dec.2019. [Online]. Available: https://itamargilad.com/the-gist-board-and-other-gist-tools/ .

[5] 4 SRE Golden Signals (What they are and why they matter), Blameless Community. 2023. [Electronic resource]. Available: https://www.blameless.com/blog/4-sre-golden-signals-what-they-are-and-why-they-matter .

[6] B. Gregg, “The USE Method,” 2023. [Electronic resource]. Available: http://www.brendangregg.com/usemethod.html .

[7] Datadog “Application Performance Monitoring (APM),” Datadog Documentation”2025. [Electronic resource]. Available: https://docs.datadoghq.com/tracing/ .

[8] Amazon Web Services, “Performance Efficiency Pillar - AWS Well-Architected Framework,” *AWS Well-Architected Documentation.* 2025. [Electronic resource]. Available: https://docs.aws.amazon.com/wellarchitected/latest/performance-efficiency-pillar/welcome.html .

[9] Ю. В. Сторожук, «Показники продуктивності програмного забезпечення інформаційних систем,» *Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення* (випуск 91). 2024. [Електронний ресурс]. Режим доступу: http://www.konferenciaonline.org.ua/ua/article/id-1905/ .

Рекомендована кафедрою програмного забезпечення ВНТУ

Стаття надійшла до редакції 10.06.2025

Сторожук Юрій Валерійович — здобувач вищої освіти третього рівня (Ph.D) факультету інформаційних технологій та комп’ютерної інженерії, e-mail: stoha27@gmail.com ;

Коваленко Олена Олексіївна — канд. техн. наук, доцент, доцент кафедри програмного забезпечення, e-mail: ок@vntu.edu.ua .

Вінницький національний технічний університет, Вінниця

Yu. V. Storozhuk[1]
O. O. Kovalenko[1]

Method for Determining Target Software Performance Indicators

[1]Vinnytsia National Technical University

Development of the methods for the determination of the target indices for measuring the quality level , performance reliability is one of the important directions in the sphere of information technologies and, in particular, software engineering.

Among the known methods we can distinguish the method of balanced management indicators, SMART-techniques, etc. GIST (Goals, Ideas, Step-Projects, Tasks) methodology it is the method of production planning, elaborated by I. Gilad former Google production manager. This methodology is aimed at reduction of the overhead cost for management, enhancement of the speed of development and creation of the products, which meet the requirements of the market and their subject area. Defining target performance indicators remains a pressing challenge for modern DevOps teams. They are frequently caught between two extremes: collecting thousands of time-series metrics, which inflates costs and alert noise, or, conversely, collecting too little data and missing performance regressions. Lightweight checklists such as the Four Golden Signals or RED/USE lower the entry barrier but remain static; they ignore business goals, service criticality, and lifecycle stage. The paper proposes to improve GIST (Goal Impact Stage Template) method for measuring software performance (GISTSP). Objective c of the study is to create and evaluate the method of target indices formation on the base of the methodology of metrics collection GIST.

The suggested GIST method (Goal – Impact – Stage – Template) relies on a four-line "service passport" and a Coreplus-Plus metric library. By following a "collect only what you need" rule, GIST automatically generates Prometheus/Grafana configurations and alert rules. The approach was experimentally compared with Four GS, RED/USE, and AWS W-A on four testbeds (Web-API, Queue, Stream, Batch). Results show that required SLI coverage increased from 78 % to 92 %; daily metric volume decreased by 60 %; setup time was reduced by a factor of four, while the F_1-score for regression detection did not degrade. These findings indicate that GIST achieves a practical balance between quick onboarding and the flexibility demanded by diverse business goals and deployment environments.

Keywords: software engineering, software testing, software performance testing, performance metrics, target performance indicators, DevOps, algorithms, templates, performance-testing knowledge bases, IT-project.

Storozhuk Yurii V. — Post-Graduate Student of the Department of Information Technologies and Computer Engineering, e-mail: stoha27@gmail.com ;

Kovalenko Olena O. — Cand. Sc. (Eng.), Associate Professor, Associate Professor of the Chair of Software, e-mail: ok@vntu.edu.ua