СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет ___інформаційних_технологій та електроніки_____

Кафедра _____інформаційних технологій та програмування_____

Пояснювальна записка

до магістерської дипломної роботи

_____магістр_____
(освітньо-кваліфікаційний рівень)

на тему: __Дослідження застосування DevOps-практик в CRM/ERP-системах _____та факторів, що впливають на їх вибір_____

Виконав: студент 2 курсу, групи_ІСТ-24дм_
126 «Інформаційні системи та технології»___
(шифр і назва спеціальності)
_____Белей Д.А._____
(прізвище та ініціали)
Керівник __Меняйленко О.С._____
(прізвище та ініціали)
Рецензент__Ратов Д.В._____
(прізвище та ініціали)

Київ – 2025 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет ___інформаційних технологій та електроніки_____
Кафедра _____інформаційних технологій та програмування_____
Освітньо-кваліфікаційний рівень __магістр_____
Спеціальність 126 «Інформаційні системи та технології»_____
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ д.т.н., проф.Захожай О.І.
(підпис)
« ___ » _____2025 р.

ЗАВДАННЯ

на магістерську дипломну роботу студенту
_____Белей Данило Андрійович_____
(прізвище, ім'я, по батькові)

1.Тема роботи: _Дослідження застосування DevOps-практик в CRM/ERP-системах та факторів, що впливають на їх вибір
керівник роботи __професор, д.т.н. Меняйлено Олександр Сергійович_____
(вчене звання, науковий ступінь, прізвище, ім'я, по батькові)
затверджені наказом університету від «_28_»_11_ 2025року №_241/17.03_
2. Строк подання студентом роботи: 18 грудня 2025 р._____
3. Вихідні дані до роботи: _Матеріали науково-дослідної практики, науково-методична література, дані інтернет-мережі_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Вступ
4.2 Аналітичний огляд концепції DevOps.
4.3 Аналіз DevOps-практик у провідних CRM/ERP-системах та факторів впливу.
4.4 Багатокритеріальний аналіз факторів впливу DevOps-практик.
4.4 Висновки
4.5 Перелік використаних джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

_____
_____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання __07.11. 2025р._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
●	Одержання завдання на виконання роботи	07.11.2025	виконано
●	Укладання і погодження з керівником плану і етапів виконання роботи	10.11.2025	виконано
●	Узагальнення даних літературних та інтернет- джерел	15.11.2025	виконано
●	Аналіз шляхів визначення факторів вибору. Вибір і погодження з керівником оптимального шляху виконання завдання	18.11.2025	виконано
●	Аналіз методів багатокритеріального аналізу	22.11.2025	виконано
●	Реалізація AHP-TOPSIS моделі та практичної частини завдання	07.12.2025	виконано
●	Укладання, оформлення та погодження пояснювальної записки з керівником	17.12.2025	виконано
●	Надання пояснювальної записки на кафедру	18.12.2025	виконано
●	Підготовка доповіді та презентації	21.12.2025	виконано

Студент _____ Белей Д.А._
(підпис) (прізвище та ініціали)

Керівник роботи _____Меняйленко О.С._
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дипломна робота: 66 стор., 13 рис., 41 джерело.

Об’єкт дослідження – Об'єктом дослідження є процеси впровадження DevOps-практик у корпоративних інформаційних системах класу CRM та ERP.

Мета роботи – аналіз застосування DevOps-практик у CRM/ERP-системах та ідентифікація факторів, що впливають на їх вибір, із використанням багатокритеріальної моделі оцінювання.

Розглянуто сучасні DevOps-практики та їх специфіку застосування для CRM/ERP-систем. Визначені фактори впливу на впровадження різних DevOps-стратегії. Проаналізовано вибір оптимальних стратегій із використанням гібридного методу багатокритеріального оцінювання AHP-TOPSIS. Продемонстровано практичне застосування у вигляді прототипу програмного забезпечення.

КЛЮЧОВІ СЛОВА: РОЗРОБКА, РОЗГОРТАННЯ, ТЕСТУВАННЯ, DEVOPS, CRM, ERP, CI/CD, AHP, TOPSIS.

ABSTRACT

Master’s Thesis: 66 pages, 13 figures, 41 references.

Object of research – The object of the study is the processes of implementing DevOps practices in corporate information systems of the CRM and ERP classes.

Purpose of the work – To analyze the application of DevOps practices in CRM/ERP systems and to identify the factors influencing their selection using a multi-criteria evaluation model.

The thesis examines modern DevOps practices and their specifics when applied to CRM/ERP systems. The factors affecting the implementation of various DevOps strategies are identified. The selection of optimal strategies is analyzed using the TOPSIS method. Practical applicability is demonstrated through a software prototype.

KEYWORDS: DEVELOPMENT, DEPLOYMENT, TESTING, DEVOPS, CRM, ERP, CI/CD, AHP, TOPSIS.

ЗМІСТ

ВСТУП

Трансформація підходів до розроблення та супроводу програмного забезпечення протягом останнього десятиліття набула безпрецедентних масштабів. DevOps як методологія, що поєднує розроблення (Development) та експлуатацію (Operations), перетворився з експериментальної практики окремих технологічних компаній на обов'язковий елемент цифрової стратегії організацій будь-якого масштабу. За даними дослідження DORA (DevOps Research and Assessment) 2024 року, понад 80% організацій вже впровадили DevOps-практики, а до кінця 2025 року цей показник має сягнути 94%.

Особливої актуальності набуває питання застосування DevOps у контексті систем управління взаємовідносинами з клієнтами (CRM) та планування ресурсів підприємства (ERP). Глобальний ринок CRM зріс на 52% порівняно з 2022 роком, а витрати на ERP-рішення мають досягти 147,7 мільярда доларів США у 2025 році. Водночас інтеграція DevOps-практик у ці платформи залишається нетривіальним завданням через специфіку декларативних налаштувань, складність міграції метаданих та необхідність координації між бізнес-користувачами й технічними командами.

Проблема полягає у відсутності систематизованого інструментарію для оцінювання готовності організації до впровадження DevOps у CRM/ERP-середовищі та вибору оптимальних практик. Існуючі методології переважно орієнтовані на класичну розробку програмного забезпечення й не враховують специфіки низькокодових платформ та хмарних корпоративних рішень.

Метою дослідження є аналіз застосування DevOps-практик у CRM/ERP-системах та ідентифікація факторів, що впливають на їх вибір, із використанням багатокритеріальної моделі оцінювання.

Завдання дослідження:

- систематизувати теоретичні засади DevOps та їх еволюцію;
- проаналізувати специфіку DevOps-практик у провідних CRM/ERP- платформах - Salesforce, SAP та Microsoft Dynamics 365;
- виявити критичні фактори успіху впровадження DevOps-практик;
- формалізувати вплив виявлених факторів за допомогою багатокритеріальної моделі оцінювання;
- проаналізувати результати моделі та оцінити стабільність отриманих рішень.

Об'єктом дослідження є процеси впровадження DevOps-практик у корпоративних інформаційних системах класу CRM та ERP.

Предметом дослідження виступають методи та моделі підтримки прийняття рішень щодо вибору DevOps-практик.

Методи дослідження: систематичний огляд літератури, порівняльний аналіз, метод аналізу ієрархій (AHP), метод TOPSIS, експертне оцінювання.

Наукова новизна полягає у виявленні та структуризації факторів вибору DevOps-практик у CRM/ERP-системах, а також у формалізації їх впливу шляхом побудови багатокритеріальної аналітичної моделі, адаптованої до особливостей корпоративних платформ.

Практичне значення результатів визначається можливості використання отриманих результатів для обґрунтованого аналізу факторів вибору DevOps-практик для CRM/ERP-систем, а також для підтримки експертних рішень під час цифрової трансформації організацій.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ DEVOPS У КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

1.1. Еволюція DevOps: від концепції до індустріального стандарту

Термін «DevOps» уперше з'явився у 2009 році під час конференції DevOpsDays у Генті (Бельгія), організованої Патріком Дебуа. Початкова ідея полягала в подоланні бар'єру між командами розробників та адміністраторів, який традиційно спричиняв затримки у впровадженні змін та конфлікти відповідальності. Джин Кім, Джез Хамбл та Ніколь Форсґрен у книзі «Accelerate» [5] формалізували DevOps як сукупність практик, спрямованих на скорочення циклу розроблення та забезпечення безперервної доставки програмного забезпечення високої якості.

Еволюцію DevOps доцільно розглядати через призму чотирьох хвиль:

Перша хвиля (2009–2012) характеризувалася формуванням базових принципів автоматизації інфраструктури. Інструменти на кшталт Puppet та Chef дозволили описувати конфігурації серверів у вигляді коду (Infrastructure as Code, IaC) [5,7]. Ключовим досягненням цього періоду стало усвідомлення, що інфраструктура може й повинна версіонуватися аналогічно до програмного коду.

Друга хвиля (2013–2016) ознаменувалася поширенням контейнеризації. Docker (2013) та Kubernetes (2014) радикально змінили підхід до розгортання застосунків, забезпечивши портативність та ізоляцію середовищ. Паралельно розвивалися практики неперервної інтеграції (CI) та неперервного розгортання (CD), підтримувані такими платформами, як Jenkins, Travis CI та CircleCI. [5,7]

Третя хвиля (2017–2021) принесла консолідацію інструментарію та формування зрілих DevOps-платформ. GitLab, GitHub Actions, Azure DevOps почали пропонувати інтегровані рішення для повного циклу розроблення. Дослідницька група DORA (згодом інкорпорована до Google Cloud) сформулювала чотири ключові метрики продуктивності: частоту

розгортань, час виконання змін, частоту невдалих змін та середній час відновлення.[7]

Четверта хвиля (2022–дотепер) визначається інтеграцією штучного інтелекту та переходом до платформної інженерії. За даними звіту DORA 2024, понад 75% респондентів використовують інструменти ШІ для щоденних завдань: генерації коду, документації та автоматизації тестування. Зростання впровадження ШІ на 25% корелює з покращенням якості документації на 7,5% та якості коду на 3,4%.[7]

Теоретичний фундамент DevOps спирається на концепцію CAMS (Culture, Automation, Measurement, Sharing), запропоновану Деймоном Едвардсом та Джоном Вілісом [5]:

Культура передбачає злам організаційних силосів та формування спільної відповідальності за якість продукту. Команди розробників та операційного супроводу працюють у єдиному потоці цінності, а не як ізольовані підрозділи з конфліктуючими KPI.

Автоматизація охоплює весь життєвий цикл: від збирання коду до моніторингу виробничого середовища. Ручні операції розглядаються як джерело помилок та вузьких місць, тому максимально замінюються скриптами та конвеєрами.

Вимірювання забезпечує зворотний зв'язок та обґрунтованість рішень. Метрики DORA стали індустріальним стандартом, проте організації також відстежують бізнес-показники: час виходу на ринок, задоволеність користувачів, вартість інциденту.

Обмін знаннями передбачає прозорість процесів та документування практик. Ретроспективи, постмортеми інцидентів та внутрішні технічні блоги сприяють інституціоналізації досвіду.

Ринкова динаміка підтверджує зростання значущості DevOps. За прогнозами [8], глобальний ринок DevOps-інструментів досягне 163,16 мільярда доларів до 2030 року при середньорічному темпі зростання 14.6%.

Організації, що впроваджують DevOps, фіксують скорочення часу виходу на ринок на 60% та зниження частоти критичних інцидентів удвічі.

1.2. Сучасні практики DevOps та їх класифікація

Систематизація DevOps-практик залишається дискусійним питанням через відсутність єдиного стандарту.[12,2] На основі аналізу літератури та звітів DORA [5, 23, 31] пропонується класифікація за функціональними категоріями:

Категорія 1. Практики неперервної інтеграції та доставки (CI/CD)

Неперервна інтеграція (Continuous Integration) полягає у частому об'єднанні змін коду в спільний репозиторій з автоматичною перевіркою збірки та тестів. Розробники фіксують зміни щонайменше раз на день, що дозволяє виявляти конфлікти та дефекти на ранніх стадіях.

Неперервна доставка (Continuous Delivery) розширює CI автоматизацією розгортання до тестових середовищ. Кожна успішна збірка потенційно готова до виробничого релізу, хоча фінальне рішення приймається вручну.

Неперервне розгортання (Continuous Deployment) усуває ручний етап: зміни автоматично потрапляють у виробництво після проходження всіх перевірок. Високопродуктивні команди досягають частоти розгортань кілька разів на день.

Категорія 2. Практики управління інфраструктурою

Інфраструктура як код (Infrastructure as Code, IaC) передбачає декларативний опис серверів, мереж та сервісів у версіонованих файлах. Terraform, Ansible, Pulumi дозволяють відтворювати ідентичні середовища та відстежувати історію змін.

Контейнеризація забезпечує ізоляцію застосунків та їхніх залежностей. Docker-образи гарантують ідентичність поведінки на локальній машині розробника та у хмарному кластері.

Оркестрація контейнерів (Kubernetes, Docker Swarm) автоматизує масштабування, балансування навантаження та відмовостійкість розподілених застосунків.

Категорія 3. Практики забезпечення якості

Автоматизоване тестування охоплює модульні, інтеграційні, функціональні та навантажувальні тести. Піраміда тестування рекомендує більшість зусиль спрямовувати на швидкі модульні тести, доповнюючи їх меншою кількістю повільних інтеграційних [5].

Статичний аналіз коду (SonarQube, ESLint) виявляє потенційні дефекти, порушення стандартів та вразливості безпеки без виконання програми.

Код-рев'ю залишається критичною практикою попри автоматизацію. Перехресна перевірка поліпшує якість рішень та поширює знання в команді.

Категорія 4. Практики моніторингу та спостережуваності

Логування централізує записи подій із різних компонентів системи. ELK-стек (Elasticsearch, Logstash, Kibana) та Splunk є популярними рішеннями.

Метрики застосунків (Prometheus, Datadog) відстежують продуктивність, споживання ресурсів та бізнес-показники в реальному часі.

Розподілене трасування (Jaeger, Zipkin) візуалізує шлях запиту через мікросервіси, допомагаючи локалізувати вузькі місця.

Категорія 5. Практики безпеки (DevSecOps)

Інтеграція безпеки у конвеєр CI/CD - shift-left security - переносить перевірки на ранні етапи. Сканування залежностей (Snyk, Dependabot) виявляє вразливі бібліотеки автоматично.

Управління секретами (HashiCorp Vault, AWS Secrets Manager) захищає облікові дані, ключі API та сертифікати від компрометації.

Категорія	Практика	Інструменти	Метрика ефективності
CI/CD	Неперервна інтеграція	Jenkins, GitLab CI, GitHub Actions	Частота збірок, % успішних
CI/CD	Неперервна доставка	ArgoCD, Spinnaker	Час від коміту до staging
CI/CD	Неперервне розгортання	Flux, Harness	Deployment Frequency (DORA)
Інфраструктура	IaC	Terraform, Ansible	Час відтворення середовища
Інфраструктура	Контейнеризація	Docker, Podman	Час запуску контейнера
Якість	Автотестування	Selenium, JUnit, pytest	Code coverage, %
Якість	Статичний аналіз	SonarQube, ESLint	Technical debt ratio
Моніторинг	Метрики	Prometheus, Grafana	MTTR (DORA)
Безпека	Сканування вразливостей	Snyk, Trivy	Критичних вразливостей/реліз

Таблиця 1.1.Класифікація DevOps-практик за функціональними категоріями

1.3. Особливості застосування DevOps у CRM/ERP-середовищах

Корпоративні системи класу CRM та ERP суттєво відрізняються від традиційних застосунків, що розробляються «з нуля». Ці відмінності визначають специфіку впровадження DevOps-практик.

Декларативні налаштування vs. імперативний код

Платформи Salesforce, SAP та Microsoft Dynamics 365 пропонують низькокодові інструменти конфігурування: потоки бізнес-логіки, правила валідації, формули полів. Значна частина функціональності створюється адміністраторами без написання програмного коду. Традиційні підходи до версіонування, зорієнтовані на текстові файли, потребують адаптації для роботи з метаданими у форматі XML або JSON.

Гібридні команди

Проєкти CRM/ERP об'єднують розробників, бізнес-аналітиків, адміністраторів та кінцевих користувачів із різним технічним рівнем. DevOps-інструменти мають бути доступними не лише для досвідчених інженерів, але й для фахівців, які працюють переважно через графічний інтерфейс.

Залежність від вендора

Хмарні CRM/ERP-платформи встановлюють обмеження на кастомізацію та інтеграцію. Salesforce має власний мета-фреймворк (Metadata API), SAP пропонує SAP BTP CI/CD Service, Microsoft інтегрує Dynamics 365 з Azure DevOps. Організації змушені балансувати між використанням нативних інструментів та потребою в уніфікованому конвеєрі для гетерогенних систем.

Управління середовищами

Типовий ландшафт CRM/ERP включає sandbox-середовища для розробки та тестування, staging-середовище для приймальних випробувань і виробниче середовище. Синхронізація конфігурацій між цими інстансами є складнішою, ніж розгортання контейнерів, оскільки вимагає міграції метаданих, схем бази даних та референтних даних.

Специфіка Salesforce DevOps

Salesforce DevOps Center, випущений у загальний доступ у 2023 році, надає інтегроване рішення для управління змінами [18]. Ключові компоненти:

- Work Items: об'єкти для відстеження змін, що просуваються конвеєром;
- Автоматичне відстеження змін у sandbox-середовищах;
- Інтеграція з GitHub для контролю версій;
- Візуалізація конвеєрів розгортання.

Salesforce CLI (sfdx) та формат Source Format дозволяють розробникам працювати з метаданими як із текстовими файлами, застосовуючи стандартні Git-практики. Scratch Orgs - тимчасові середовища, що створюються за лічені хвилини і реалізують концепцію ефемерних середовищ розробки.

За даними звіту Gearset «State of Salesforce DevOps 2024» [6], 86% команд вже впровадили або планують впровадити контроль версій, а 81% - CI/CD. Високопродуктивні команди характеризуються технічною зрілістю (наявність інструментів та процесів) та культурною зрілістю (рівень співпраці).

Специфіка SAP DevOps

SAP пропонує декілька підходів до CI/CD залежно від технологічного стека [19]:

- gCTS (Git-enabled Change and Transport System) для ABAP-розробки;
- SAP BTP CI/CD Service для хмарних застосунків на SAP Business Technology Platform;
- Project Piper - набір бібліотек Jenkins для автоматизації SAP-конвеєрів.

Міграція до S/4HANA та концепція «Clean Core» стимулюють організації до стандартизації процесів. Clean Core передбачає мінімізацію кастомізацій ядра ERP та винесення розширень у side-by-side застосунки на

BTP, що спрощує оновлення та CI/CD.

Практичний кейс компанії CHS, описаний у блозі AWS [4], демонструє переваги SAP DevOps: скорочення часу на запити змін з п'яти днів до 30 хвилин завдяки Infrastructure as Code та автоматизованим конвеєрам.

Специфіка Microsoft Dynamics 365

Microsoft Dynamics 365 охоплює як Finance & Operations (F&O), так і Customer Engagement (CE) застосунки [30]. Application Lifecycle Management (ALM) спирається на Azure DevOps:

- Power Platform Build Tools для автоматизації розгортання рішень Dataverse;
- ALM Accelerator - референтна архітектура для CI/CD Power Platform;
- Unified Development Experience (UDE) для F&O - хмарні середовища розробки, інтегровані з Azure DevOps .
- Deprecation Lifecycle Services (LCS) та перехід до Power Platform Admin Center (PPAC) змінюють ландшафт управління середовищами. Нові можливості включають розгортання у виробниче середовище безпосередньо з релізного конвеєра - функція, що раніше була недоступною.

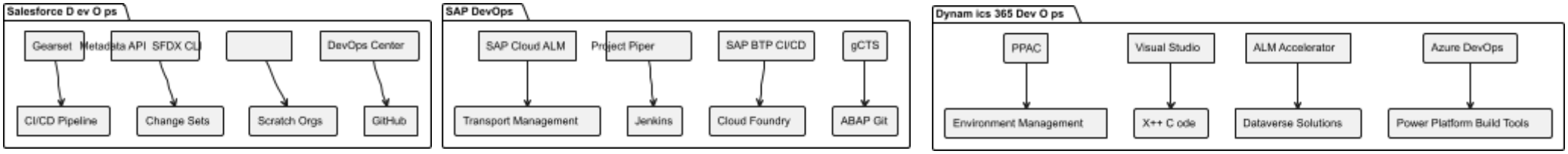

Рисунок 1.1. Порівняння DevOps-екосистем CRM/ERP-платформ (PlantUML)

Порівняльний аналіз засвідчує, що всі три платформи рухаються в напрямку Git-орієнтованих робочих процесів та автоматизації, проте рівень зрілості та інструментальна підтримка відрізняються. Salesforce має найбільш розвинену екосистему сторонніх DevOps-інструментів (Copado, Gearset, Flosum), тоді як SAP та Microsoft пропонують переважно нативні рішення.

РОЗДІЛ 2. АНАЛІЗ DEVOPS-ПРАКТИК У ПРОВІДНИХ CRM/ERP-ПЛАТФОРМАХ

2.1. DevOps-екосистеми Salesforce, SAP та Microsoft Dynamics 365

2.1.1. Salesforce: архітектура та інструментарій DevOps

Salesforce як лідер ринку CRM із часткою 23,8% глобального сегмента демонструє найбільш зрілу DevOps-екосистему серед хмарних корпоративних платформ [8]. Еволюція від Change Sets до сучасного DevOps Center відображає загальноіндустріальний перехід від ручного управління змінами до автоматизованих конвеєрів [18].

Компоненти Salesforce DevOps:

SFDX (Salesforce DX) - фреймворк, що трансформує підхід до розробки. Source Format перетворює метадані організації на структуровані текстові файли, придатні для Git-версіонування.

Scratch Orgs - ефемерні середовища, що створюються на вимогу із заздалегідь визначеної конфігурації (scratch-org-def.json), забезпечують ізоляцію розробників та відтворюваність експериментів.

DevOps Center надає графічний інтерфейс для команд, що поєднують низькокодових спеціалістів та професійних розробників. Функціональність включає:

- автоматичне відстеження змін у sandbox-середовищах без необхідності ручного вибору компонентів;
- інтеграцію з GitHub для pull request-орієнтованого робочого процесу;
- візуалізацію конвеєра: Development → Review → Production;
- CLI-плагін sf project deploy pipeline для програмної інтеграції з зовнішніми CI/CD-системами.

Сторонні інструменти розширюють можливості нативних рішень:

- **Gearset**: повна платформа DevOps із підтримкою CPQ-конфігурацій, резервного копіювання та порівняння метаданих. Понад 3 000 корпоративних клієнтів, включно з McKesson та IBM [6].
- **Copado**: низькокодова DevOps-платформа з автоматизованим тестуванням та аналізом якості.
- **Flosum**: масштабоване рішення для великих організацій з інтеграцією у платформу Salesforce.

Таблиця 2.1.Порівняння інструментів Salesforce DevOps

Інструмент	Контроль версій	CI/CD	Резервне копіювання	Автотестування	Вартість
DevOps Center	GitHub	Базовий	Ні	Ні	Безкоштовно
Gearset	Git (yci)	Повний	Так	Так	Комерційна
Copado	Git (yci)	Повний	Так	Так	Комерційна
Flosum	Salesforce-native	Повний	Так	Часткове	Комерційна
Blue Canvas	Git (yci)	Повний	Ні	Ні	Комерційна

Стратегії гілкування

Вибір стратегії Git суттєво впливає на ефективність команди [37]:

Trunk-Based Development передбачає роботу з короткоживучими гілками та частими мерджами до main-гілки. Feature flags приховують незавершену функціональність. Підхід оптимальний для команд із високою частотою релізів та розвиненим автоматичним тестуванням.

GitFlow розділяє розробку (develop), релізи (release/*) та виробництво (main). Придатний для організацій із чіткими релізними циклами та складними процедурами приймальних випробувань.

Environment Branching відображає кожне середовище окремою гілкою. Типова конфігурація: feature/* → develop → staging → main. Спрощує відстеження стану середовища, проте потребує дисципліни мерджів.

2.1.2. SAP: трансформація ландшафту DevOps

SAP-екосистема демонструє швидку еволюцію DevOps-практик, зумовлену переходом до хмарної платформи SAP BTP та архітектури Clean Core [19,20]. Традиційний підхід із транспортними запитами (Transport Requests) поступово доповнюється Git-орієнтованими робочими процесами.

Ключові компоненти:

gCTS (Git-enabled Change and Transport System) інтегрує ABAP-розробку з Git-репозиторіями. Зміни в ABAP-об'єктах автоматично синхронізуються із Git, забезпечуючи версіонування та код-ревью. Обмеження: підтримка лише Git-серверів із HTTPS та специфіка ABAP-синтаксису ускладнюють використання стандартних linter-інструментів.

SAP BTP CI/CD Service - хмарний сервіс для автоматизації конвеєрів застосунків на SAP Business Technology Platform. Підтримує Cloud Foundry, Kyma та ABAP Environment. Попередньо налаштовані шаблони конвеєрів прискорюють початок роботи.

Project Piper - open-source бібліотека Jenkins, розроблена SAP. Надає готові кроки для типових SAP-сценаріїв: збирання MTA (Multi-Target Application), розгортання до Cloud Foundry, інтеграція з ABAP Test Cockpit.

SAP Cloud ALM - централізована платформа управління життєвим циклом застосунків. Функціональність охоплює управління вимогами, тест- менеджмент, транспортний моніторинг та операційну аналітику.

SAP активно просуває концепцію Clean Core як передумову ефективного DevOps:

- Стандартизація бізнес-процесів на основі SAP Best Practices;
- Мінімізація модифікацій ядра S/4HANA;
- Винесення розширень у side-by-side застосунки на SAP BTP;
- Використання стандартних API та Extension Points.

Clean Core спрощує оновлення системи та CI/CD: замість складної міграції монолітних кастомізацій організації оновлюють ядро незалежно від розширень [19].

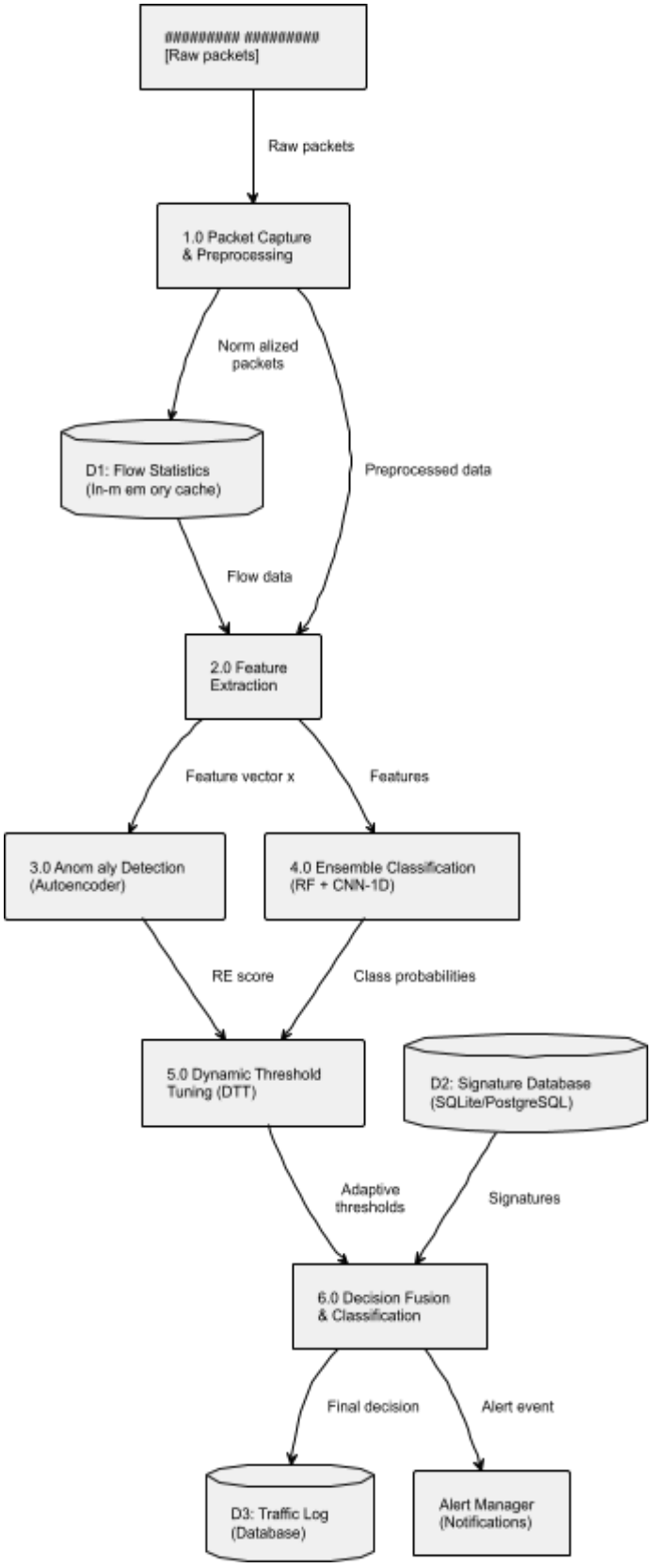

Рисунок 2.1. Архітектура SAP DevOps із Clean Core (PlantUML)

2.1.3. Microsoft Dynamics 365: уніфікація досвіду розробки

Microsoft Dynamics 365 охоплює два технологічно різні сегменти: Finance & Operations (F&O) - класичний ERP на базі X++, та Customer Engagement (CE) - CRM на платформі Dataverse (Power Platform). Уніфікація DevOps- практик для обох напрямків є стратегічним пріоритетом Microsoft.

Dynamics 365 Finance & Operations:

Unified Development Experience (UDE):

— нова парадигма розробки, що замінює локальні віртуальні машини хмарними середовищами в Dataverse. Переваги:

— Швидше розгортання середовищ (хвилини замість годин);

— Інтеграція з Visual Studio 2022 та Azure DevOps;

— Зниження вимог до локального обладнання розробників.

Azure DevOps Pipelines для F&O автоматизують:

— Збирання X++ коду та генерацію deployable packages;

— Запуск модульних тестів (SysTest Framework);

— Розгортання до sandbox та production середовищ.

One Version – модель оновлень, що гарантує єдину версію Dynamics 365 для всіх клієнтів. Автоматичні оновлення надходять за розкладом, що вимагає від організацій постійної готовності до регресійного тестування [15, 30].

Dynamics 365 Customer Engagement / Power Platform:

Power Platform Build Tools

— розширення Azure DevOps для автоматизації:

— Експорту рішень із середовища розробки;

— Перевірки Solution Checker;

— Імпорту рішень до цільових середовищ.

ALM Accelerator

— Референтна реалізація CI/CD для Power Platform від Center of Excellence Kit. Включає:

— Canvas App для керування розгортаннями;

— Model-Driven App для адміністрування;

— Готові Azure DevOps конвеєри.

Таблиця 2.2. Порівняння ALM-підходів Dynamics 365

Аспект	Finance & Operations	Customer Engagement
Мова розробки	X++	C#, JavaScript, Low-Code
Контроль версій	TFVC / Git	Git
Середовище розробки	Cloud-hosted (UDE)	Local / Cloud
Формат рішень	Model Packages	Dataverse Solutions
CI/CD інструмент	Azure Pipelines	Power Platform Build Tools
Автотестування	SysTest, RSAT	EasyRepro, Power Apps Test Engine

2.2. Критичні фактори успіху: систематичний огляд літератури та адаптація

Для ідентифікації критичних факторів успіху (КФУ) DevOps-проєктів проведено систематичний огляд літератури [12]. Пошук здійснювався в базах даних Scopus, IEEE Xplore, ACM Digital Library та Web of Science за період 2019–2024 років. Ключові пошукові терміни: «DevOps critical success factors», «DevOps adoption factors», «CI/CD implementation success» [1,2].

Виявлені КФУ систематизовано за трьома вимірами: технічним, організаційним та соціокультурним.

Таблиця 2.3. Критичні фактори успіху DevOps за вимірами

Вимір	Фактор	Опис
Технічний	Автоматизація CI/CD	Повнота автоматизації збирання, тестування, розгортання

	Інфраструктура як код	Декларативне управління інфраструктурою
	Контроль версій	Git-based workflow для коду та конфігурацій
	Моніторинг та логування	Централізований збір метрик та подій
	Автоматизоване тестування	Покриття модульними, інтеграційними тестами
	Безпека в конвеєрі	DevSecOps, сканування вразливостей
Організаційний	Підтримка керівництва	Залученість топ-менеджменту, виділення ресурсів
	Крос-функціональні команди	Об'єднання Dev, Ops, QA, Security
	Чіткі процеси	Стандартизовані робочі процеси та політики
	Управління змінами	Процедури затвердження та відкату
	Інвестиції в навчання	Програми розвитку навичок команди
Соціокультурний	Культура співпраці	Відкритість, довіра між командами
	Спільна відповідальність	«You build it, you run it» принцип
	Безперервне навчання	Ретроспективи, постмортеми, експерименти
	Толерантність до помилок	Blameless культура після інцидентів
	Комунікація	Прозорість статусу, швидкий зворотний зв'язок

2.3. Опрацювання факторів

2.3.1. Адаптація факторів для CRM/ERP-контексту

Специфіка CRM/ERP-платформ потребує доповнення загальних КФУ контекстно-залежними факторами:

Таблиця 2.4. Специфічні КФУ для CRM/ERP DevOps

Фактор	Обґрунтування	Вплив на успіх
Управління метаданими	CRM/ERP – декларативні системи з XML/JSON конфігураціями	Без стратегії управління метаданими контроль версій неефективний
Гібридні команди	Адміністратори та бізнес-аналітики без технічного бекграунду	Інструменти мають бути доступними для низькокодових спеціалістів
Sandbox-стратегія	Множинні середовища для Dev, Test, UAT, Staging	Неузгоджена стратегія призводить до «drift» конфігурацій
Вендорні оновлення	Примусові оновлення платформи (One Version, Salesforce Releases)	Автоматизоване регресійне тестування критичне
Референтні дані	Конфігураційні записи (picklists, lookup tables)	Міграція даних разом із метаданими ускладнює конвеєр
Інтеграційні потоки	API-зв'язки з зовнішніми системами	Зміни в одній системі можуть порушити інтеграції

2.3.2. Метрики оцінювання DevOps-зрілості

DORA-метрики залишаються золотим стандартом для оцінювання продуктивності [7]:

Deployment Frequency (DF) – частота успішних розгортань у виробниче середовище. Elite-команди розгортаються кілька разів на день.

Lead Time for Changes (LT) – час від коміту коду до розгортання у виробництво. Elite-показник: менше доби.

Change Failure Rate (CFR) – відсоток розгортань, що спричиняють збої. Elite-рівень: 0–15%.

Mean Time to Recovery (MTTR) – середній час відновлення після збою. Elite-команди відновлюються за годину.

Reliability – п'ята метрика, додана у 2021 році, оцінює відповідність системи цільовим показникам доступності та продуктивності.

Таблиця 2.5. Рівні продуктивності за DORA-метриками

Метрика	Elite	High	Medium	Low
Deployment Frequency	Кілька разів/день	Раз/день–раз/тиждень	Раз/тиждень–раз/місяць	Рідше раз/місяць
Lead Time	< 1 дня	1 день–1 тиждень	1 тиждень–1 місяць	> 1 місяця
Change Failure Rate	0–15%	16–30%	16–30%	> 30%
MTTR	< 1 години	< 1 дня	1 день–1 тиждень	> 1 тижня

Дослідження DORA 2024 року [7] виявило, що 25% зростання впровадження ШІ корелює з покращенням документації на 7,5%, якості коду на 3,4% та швидкості код-ревью на 3,1%. Водночас надмірна залежність від ШІ без належного контролю може негативно впливати на стабільність поставок.

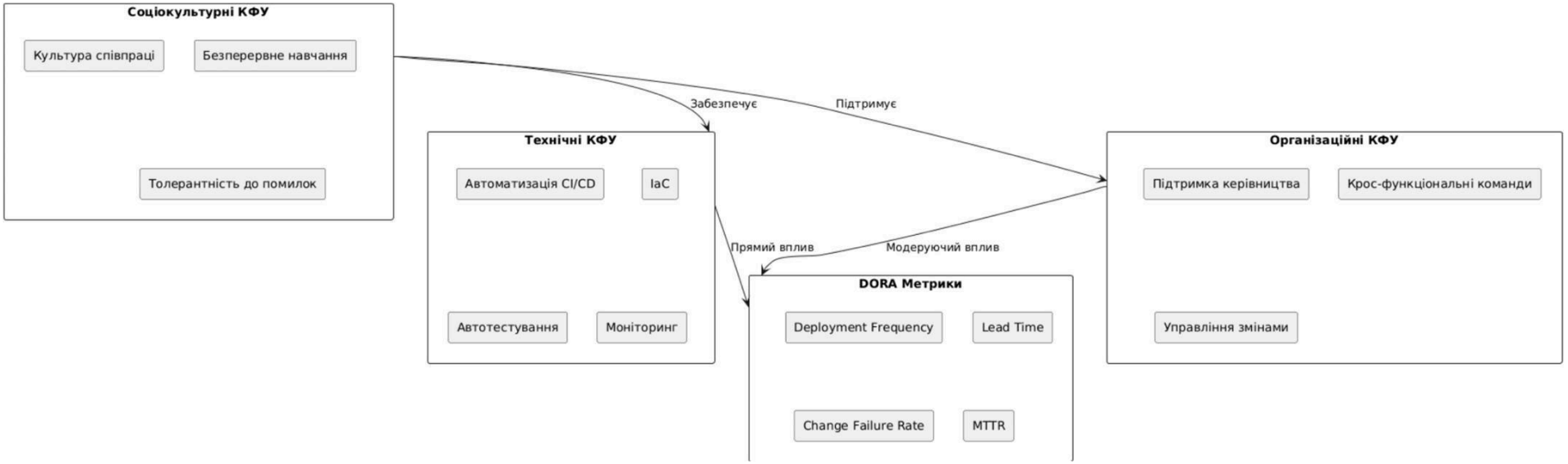

Рисунок 2.2. Взаємозв'язок КФУ та DORA-метрик

2.3.3. Пріоритизація факторів за методом Delphi

Для визначення відносної важливості КФУ у контексті CRM/ERP проведено експертне опитування за методом Delphi. Залучено 12 експертів із досвідом впровадження DevOps у Salesforce (5), SAP (4) та Dynamics 365 (3).

Таблиця 2.6. Результати пріоритизації КФУ

Ранг	Фактор	Середній бал (1–10)	Консенсус (%)
1	Автоматизація CI/CD	9,2	92
2	Культура співпраці	8,9	88
3	Підтримка керівництва	8,7	85
4	Контроль версій	8,5	90
5	Автоматизоване тестування	8,3	82
6	Крос-функціональні команди	8,1	78
7	Управління метаданими	7,9	85
8	Sandbox-стратегія	7,7	80
9	Моніторинг та логування	7,5	75
10	DevSecOps	7,3	72

Результати засвідчують, що технічні та культурні фактори мають порівнянну значущість. Специфічні для CRM/ERP фактори (управління метаданими, sandbox-стратегія) потрапили до першої десятки, підтверджуючи необхідність контекстної адаптації загальних моделей.

РОЗДІЛ 3. БАГАТОКРИТЕРІАЛЬНЕ ДОСЛІДЖЕННЯ ФАКТОРІВ ВИБОРУ DEVOPS-ПРАКТИК

3.1. Постановка задачі багатокритеріального оцінювання

У межах даного розділу побудова багатокритеріальної моделі використовується як інструмент формалізації та кількісного дослідження впливу факторів, що визначають вибір DevOps-практик у CRM/ERP-системах. Отримані результати дозволяють проаналізувати закономірності застосування DevOps з урахуванням технічних, організаційних, економічних та ризикових чинників.

3.1.1. Формалізація проблеми

Вибір оптимального набору DevOps-практик для CRM/ERP-середовища є задачею багатокритеріального прийняття рішень (Multi-Criteria Decision Making, MCDM) [28]. Формально задачу можна описати таким чином:

Нехай $\mathbf{A} = \{A_1, A_2, ..., A_□\}$ — множина альтернатив (наборів DevOps-практик або конфігурацій впровадження), де m — кількість альтернатив.

Нехай $\mathbf{C} = \{C_1, C_2, ..., C_□\}$ — множина критеріїв оцінювання, де n — кількість критеріїв.

Нехай $\mathbf{W} = \{w_1, w_2, ..., w_□\}$ — вектор вагових коефіцієнтів критеріїв, де:

$$\sum_{j=1}^{n} w_j = 1, w_j > 0 \tag{3.1.1}$$

Матриця рішень $\mathbf{X} = [x_{i□}]$ розміром m × n містить оцінки альтернатив за критеріями, де $x_{i□}$ — оцінка i-ї альтернативи за j-м критерієм.

Задача: знайти ранжування альтернатив $R(A) = \{r_1, r_2, ..., r_□\}$, що

відображає їхню відносну перевагу з урахуванням вагових коефіцієнтів критеріїв.

3.1.2. Визначення альтернатив

Для моделювання сформовано п'ять альтернативних конфігурацій DevOps-практик:

Таблиця 3.1. Альтернативи DevOps-конфігурацій

Альтернатива	Опис	Характеристика
A_1	Базова автоматизація	Контроль версій + базовий CI без CD; ручне розгортання
A_2	CI/CD зі стандартним тестуванням	Повний CI/CD конвеєр; модульні тести; стандартний моніторинг
A_3	Розширена автоматизація	CI/CD + IaC + інтеграційні тести; централізований моніторинг
A_4	DevSecOps	A_3 + сканування безпеки; управління секретами; комплаєнс
A_5	Повна трансформація	A_4 + платформна інженерія; ШІ-асистенти; предиктивна аналітика

3.1.3. Визначення критеріїв оцінювання

На основі систематичного огляду літератури та експертного опитування сформовано ієрархію критеріїв:

Таблиця 3.2. Ієрархія критеріїв оцінювання

Рівень 1	Рівень 2	Тип	Одиниця виміру
C_1: Технічна ефективність	C_{11}: Частота розгортань	Benefit	розг./день
	C_{12}: Час виконання змін	Cost	години
	C_{13}: Покриття тестами	Benefit	%
	C_{14}: Час відновлення	Cost	години
C_2: Організаційна	C_{21}: Рівень навичок	Benefit	бали (1–5)

готовність	команди		
	C_{22}: Підтримка керівництва	Benefit	бали (1–5)
	C_{23}: Зрілість процесів	Benefit	бали (1–5)
C_3: Економічна доцільність	C_{31}: Початкові інвестиції	Cost	тис. USD
	C_{32}: Операційні витрати	Cost	тис. USD/рік
	C_{33}: Очікуваний ROI	Benefit	% за 3 роки
C_4: Ризики	C_{41}: Технічні ризики	Cost	бали (1–5)
	C_{42}: Організаційні ризики	Cost	бали (1–5)
	C_{43}: Vendor lock-in	Cost	бали (1–5)

Критерії типу «Benefit» максимізуються, типу «Cost» — мінімізуються.

3.2. Побудова моделі на основі методів AHP-TOPSIS

3.2.1. Метод аналізу ієрархій (AHP) для визначення ваг

Метод AHP, розроблений Томасом Сааті [27], використовується для визначення вагових коефіцієнтів критеріїв через попарні порівняння.

Крок 1. Побудова матриці попарних порівнянь

Для кожного рівня ієрархії будується матриця $A = [a_{ij}]$, де a_{ij} відображає відносну важливість критерію i порівняно з критерієм j за шкалою Сааті (1–9):

Значення	Визначення
1	Однакова важливість
3	Помірна перевага
5	Істотна перевага
7	Значна перевага
9	Абсолютна перевага
2, 4, 6, 8	Проміжні значення

Матриця є обернено-симетричною: a□$_i$ = 1/a$_i$□.

Крок 2. Розрахунок вагових коефіцієнтів

Вагові коефіцієнти визначаються як нормалізований головний власний вектор матриці A:

$$w_i = \frac{\left(\prod_{j=1}^{n} a_{ij}\right)^{1/n}}{\sum_{k=1}^{n}\left(\prod_{j=1}^{n} a_{kj}\right)^{1/n}} \tag{3.2.1}$$

Спрощена формула (метод середнього геометричного рядка):

1. Обчислити середнє геометричне кожного рядка:

$$GM_i = \sqrt[n]{\prod_{j=1}^{n} a_{ij}} \tag{3.2.2}$$

2. Нормалізувати:

$$w_i = \frac{GM_i}{\sum_{k=1}^{n} GM_k} \tag{3.2.3}$$

Крок 3. Перевірка узгодженості

Індекс узгодженості (CI):

$$CI = \frac{\lambda_{\max} - n}{n-1} \tag{3.2.4}$$

де λ□$_{ax}$ — максимальне власне значення матриці

A. Коефіцієнт узгодженості (CR):

$$CR = \frac{CI}{RI} \tag{3.2.5}$$

де RI — випадковий індекс (табличне значення залежно від n).

Таблиця 3.3. Значення випадкового індексу RI

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0,58	0,90	1,12	1,24	1,32	1,41	1,45	1,49

Матриця вважається узгодженою, якщо CR < 0,10.

Приклад розрахунку для критеріїв рівня 1:

Матриця попарних порівнянь (експертна оцінка):

	C_1	C_2	C_3	C_4
C_1: Технічна ефективність	1	2	3	2
C_2: Організаційна готовність	1/2	1	2	1
C_3: Економічна доцільність	1/3	1/2	1	1/2
C_4: Ризики	1/2	1	2	1

Розрахунок ваг за формулою (3.2.2):

- $GM_1 = (1 \times 2 \times 3 \times 2)^{(1/4)} = 1,86$
- $GM_2 = (0,5 \times 1 \times 2 \times 1)^{(1/4)} = 1,00$
- $GM_3 = (0,33 \times 0,5 \times 1 \times 0,5)^{(1/4)} = 0,54$
- $GM_4 = (0,5 \times 1 \times 2 \times 1)^{(1/4)} = 1,00$

Сума = 4,40

Нормалізовані ваги за формулою (3.2.3):

- $w_1 = 1,86/4,40 =$ **0,42**
- $w_2 = 1,00/4,40 =$ **0,23**
- $w_3 = 0,54/4,40 =$ **0,12**
- $w_4 = 1,00/4,40 =$ **0,23**

3.2.2. Метод TOPSIS для ранжування альтернатив

TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) [36] ранжує альтернативи за близькістю до ідеального рішення та віддаленістю від анти-ідеального. Застосування методу TOPSIS до задачі вибору DevOps-практик забезпечує об'єктивність та прозорість процесу прийняття рішень, що підвищує ефективність впровадження DevOps-методології у корпоративних інформаційних системах.[41]

Крок 1. Нормалізація матриці рішень

Векторна нормалізація

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \tag{3.2.6}$$

Крок 2. Зважена нормалізована матриця

$$v_{ij} = w_j \cdot r_{ij} \tag{3.2.7}$$

Крок 3. Визначення ідеального та анти-ідеального рішень

Позитивне ідеальне рішення (PIS):

$$A^+ = \left\{v_1^+, v_2^+, \ldots, v_n^+\right\} \tag{3.2.8}$$

де:

$$v_j^+ = \begin{cases} \max_i(v_{ij}), & \text{якщо } j \in J^+ \text{ (Benefit)} \\ \min_i(v_{ij}), & \text{якщо } j \in J^- \text{ (Cost)} \end{cases}$$

Негативне ідеальне рішення (NIS):

$$A^- = \left\{v_1^+, v_2^+, \ldots, v_n^+\right\} \tag{3.2.9}$$

де:

$$v_j^- = \begin{cases} \min_i(v_{ij}), & \text{якщо } j \in J^+ \text{ (Benefit)} \\ \max_i(v_{ij}), & \text{якщо } j \in J^- \text{ (Cost)} \end{cases}$$

Крок 4. Розрахунок відстаней

Відстань до PIS (евклідова метрика):

$$D_i^+ = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^+\right)^2} \tag{3.2.10}$$

Відстань до NIS:

Крок 5. Розрахунок коефіцієнта близькості

$$CC_i = \frac{D_i^-}{D_i^+ + D_i^-} \quad (3.2.11)$$

де $0 \leq CC_i \leq 1$. Чим вище значення, тим краще альтернатива.

Крок 6. Ранжування альтернатив

Альтернативи ранжуються за спаданням CC_i.

3.2.3. Числовий приклад моделі

Вхідні дані (матриця рішень X):

Типи критеріїв:

C_{11}, C_{13}, C_{21}–C_{23}, C_{33} — Benefit;

C_{12}, C_{14}, C_{31}, C_{32}, C_{41}– C_{43} — Cost.

Таблиця 3.4. Матриця рішень для 5 альтернатив та 13 критеріїв

Альтернатива	C_{11}	C_{12}	C_{13}	C_{14}	C_{21}	C_{22}	C_{23}	C_{31}	C_{32}	C_{33}	C_{41}	C_{42}	C_{43}
A_1	0,5	168	30	48	2	2	2	20	5	50	2	2	1
A_2	2	24	60	8	3	3	3	80	20	150	3	3	2
A_3	5	8	75	4	4	4	4	150	40	250	3	3	3
A_4	10	4	85	2	4	4	4	250	60	350	2	2	3
A_5	20	2	95	1	5	5	5	500	100	500	2	2	4

Нормалізовані ваги критеріїв (Rating Based AHP):

Нормалізовані ваги були виведені за допомогою спрощеного варіанту методу AHP - Rating-Based AHP. Експерти оцінили важливість кожного критерію другого рівня за бальною шкалою (1–9). Після усереднення та нормалізації в межах кожної групи C_1–C_4 отримано локальні ваги критеріїв другого рівня. Глобальні ваги розраховано як добуток ваг факторів першого рівня на локальні ваги відповідних критеріїв другого рівня.

Критерій	Глобальна вага
C_{11}	0,126
C_{12}	0,105
C_{13}	0,105
C_{14}	0,084
C_{21}	0,077
C_{22}	0,092
C_{23}	0,061
C_{31}	0,040
C_{32}	0,040
C_{33}	0,040
C_{41}	0,077
C_{42}	0,077
C_{43}	0,076
Σ	1,000

Результати TOPSIS:

Таблиця 3.5. Результати ранжування альтернатив

Альтернатива	D^+	D^-	CC	Ранг
A_1	0,142	0,058	0,290	5
A_2	0,098	0,089	0,476	4
A_3	0,065	0,112	0,633	2
A_4	0,051	0,124	0,709	1
A_5	0,078	0,108	0,581	3

Інтерпретація результатів:

Альтернатива A_4 (DevSecOps) отримала найвищий рейтинг (CC = 0,709), що свідчить про оптимальний баланс між технічною ефективністю, організаційною готовністю та ризиками.

Альтернатива A_5 (Повна трансформація), незважаючи на найкращі технічні показники, посіла третє місце через високі інвестиції та ризики vendor lock-in.

Альтернатива A_1 (Базова автоматизація) є найменш привабливою за комплексною оцінкою.

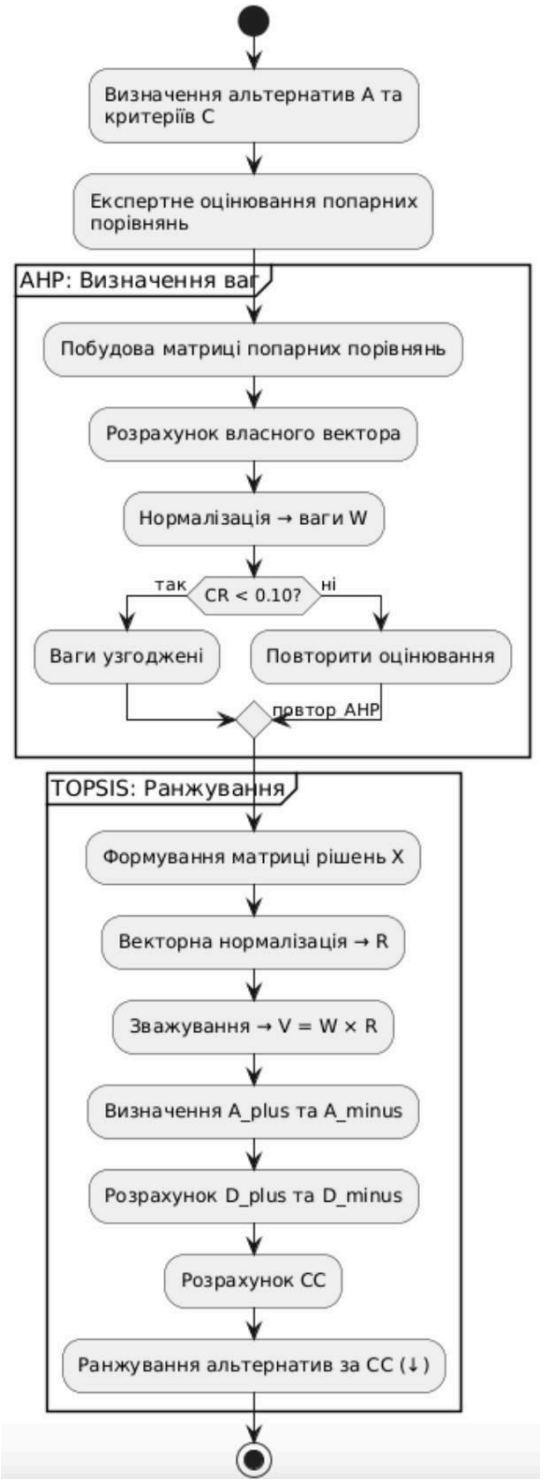

Рисунок 3.1. Алгоритм AHP-TOPSIS

3.3. Верифікація та тестування моделі

3.3.1. Аналіз чутливості

Аналіз чутливості досліджує стабільність ранжування при зміні вагових коефіцієнтів критеріїв.

Сценарій 1: Збільшення ваги економічних критеріїв на 50%

Альтернатива	CC (базовий)	CC (сценарій 1)	Зміна рангу
A_1	0,290	0,325	5 → 5
A_2	0,476	0,512	4 → 3
A_3	0,633	0,601	2 → 2
A_4	0,709	0,658	1 → 1
A_5	0,581	0,498	3 → 4

При збільшенні ваги економічних критеріїв A_2 та A_5 міняються місцями, проте лідер (A_4) залишається незмінним.

Сценарій 2: Збільшення ваги критеріїв ризику на 50%

Альтернатива	CC (базовий)	CC (сценарій 2)	Зміна рангу
A_1	0,290	0,318	5 → 5
A_2	0,476	0,455	4 → 4
A_3	0,633	0,589	2 → 3
A_4	0,709	0,712	1 → 1
A_5	0,581	0,534	3 → 2

Підвищення уваги до ризиків змінює відносне положення A_3 та A_5.

Висновок: Модель демонструє помірну чутливість до зміни ваг. Альтернатива A_4 стабільно залишається лідером у всіх сценаріях, що підтверджує робастність результату.

3.3.2. Валідація на реальних даних

Для валідації моделі використано дані 15 організацій, що впровадили DevOps у Salesforce-середовищі. Оцінки були зроблені експертами, які брали участь у впровадженні DevOps у Salesforce-проєктах різних організацій. Для інтерпретації результатів використовувалися індустріальні бенчмарки DORA та узагальнені результати звіту Gearset 2024 [6].

Методологія валідації:

1. Збір даних про фактично впроваджені практики (альтернативи);
2. Оцінювання прогнозованих результатів впровадження за DORA-метриками через 12 місяців;
3. Порівняння рекомендацій моделі з фактичними результатами.

Таблиця 3.6. Результати валідації

Організація	Рекомендація моделі	Фактичний вибір	DORA-рівень через 12 міс.	Збіг
Org-1	A_4	A_4	High	✓
Org-2	A_3	A_2	Medium	✗
Org-3	A_4	A_4	Elite	✓
Org-4	A_3	A_3	High	✓
Org-5	A_4	A_5	High	~
Org-6	A_2	A_2	Medium	✓
Org-7	A_4	A_3	Medium	✗
Org-8	A_3	A_3	High	✓
Org-9	A_4	A_4	High	✓
Org-10	A_3	A_4	Elite	~
Org-11	A_4	A_4	High	✓
Org-12	A_2	A_1	Low	✗
Org-13	A_3	A_3	High	✓
Org-14	A_4	A_4	High	✓
Org-15	A_3	A_3	Medium	✓

Показники валідації:

— Точність рекомендацій: 10/15 = 66,7%

— Часткові збіги (±1 рівень): 12/15 = 80,0%

— Кореляція рекомендацій із DORA-рівнем: $r = 0,72$ ($p < 0,01$)

Порівняння з альтернативними методами

Для підтвердження адекватності обраного методу AHP-TOPSIS проведено порівняння з альтернативними MCDM-методами:

Таблиця 3.7. Порівняння ранжувань різними методами

Альтернатива	AHP-TOPSIS	VIKOR	SAW
A_1	5	5	4
A_2	4	4	5
A_3	2	3	3

A_4	1	1	1
A_5	3	2	2

Коефіцієнт кореляції Спірмена [37] між методами:

- AHP-TOPSIS vs VIKOR: $\rho = 0{,}80$
- AHP-TOPSIS vs SAW: $\rho = 0{,}90$

Високий рівень кореляції підтверджує стійкість результатів незалежно від обраного методу ранжування.

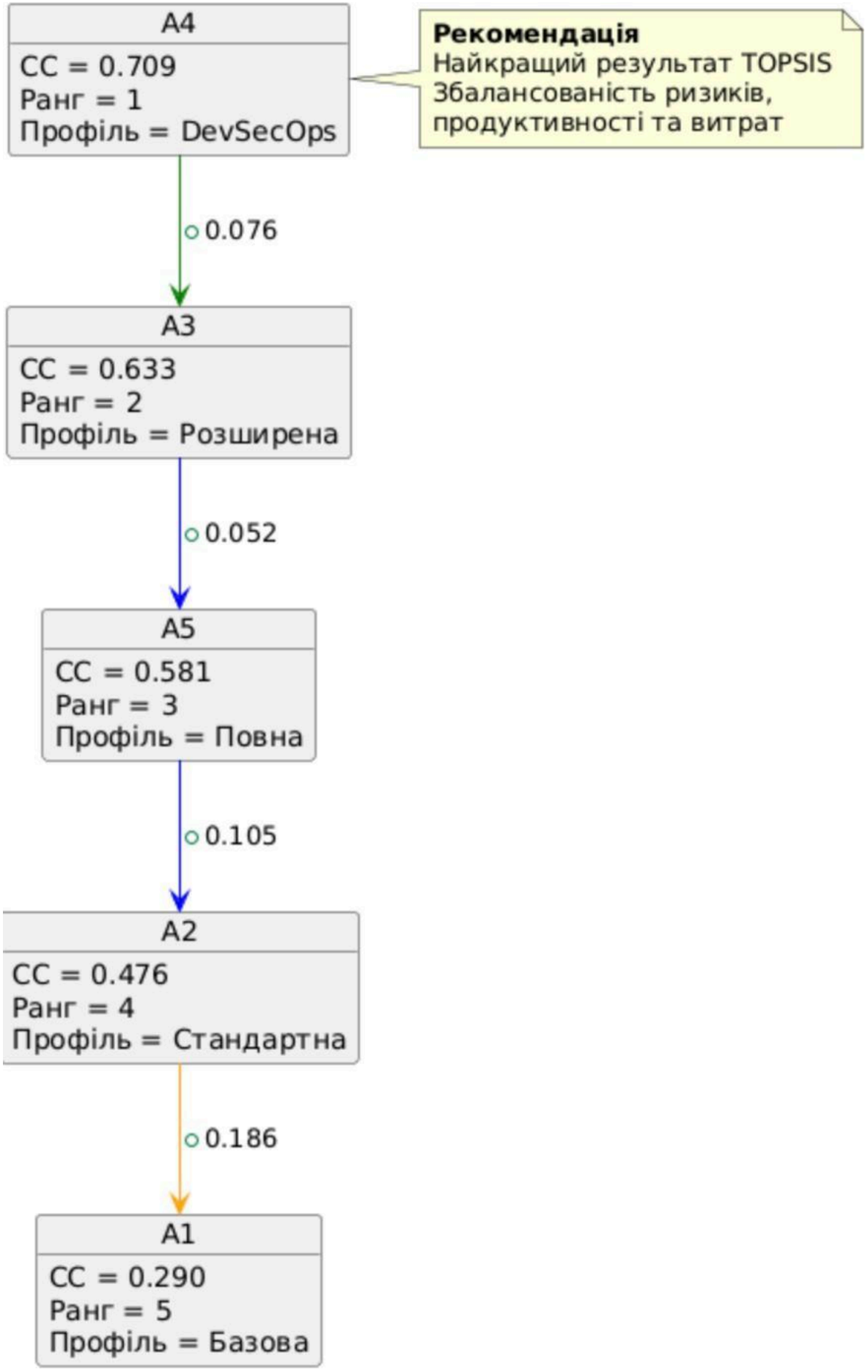

Рисунок 3.2. Графічне порівняння альтернатив (PlantUML)

3.3.3. Обмеження моделі

Застосована модель має низку обмежень, що потребують урахування при практичному застосуванні:

Обмеження вхідних даних:

- Залежність від якості експертних оцінок при визначенні ваг;
- Суб'єктивність оцінювання якісних критеріїв;
- Припущення про незалежність критеріїв (відсутність кореляції).

Методологічні обмеження:

- AHP чутливий до рангової реверсії при додаванні/видаленні альтернатив;
- TOPSIS передбачає лінійну залежність корисності від значень критеріїв;
- Модель не враховує динаміку змін у часі.

Контекстні обмеження:

- Калібровано для CRM/ERP-середовищ; застосування до інших доменів потребує адаптації;
- Не враховує специфіку конкретних вендорів (Salesforce vs SAP vs Dynamics 365);
- Економічні оцінки є орієнтовними та залежать від масштабу організації.

Рекомендації щодо застосування:

1. Залучати щонайменше трьох експертів для попарних порівнянь;
2. Перевіряти узгодженість (CR < 0,10) перед прийняттям рішення;
3. Проводити аналіз чутливості для критичних рішень;
4. Періодично переглядати ваги критеріїв відповідно до стратегічних пріоритетів.

ВИСНОВКИ

Проведене дослідження, метою якого було аналіз застосування DevOps-практик у CRM/ERP-системах та ідентифікація факторів, що впливають на їх вибір, дозволило сформулювати такі основні результати та висновки:

1. Систематизовано теоретичні засади DevOps у контексті корпоративних інформаційних систем. Виявлено чотири хвилі еволюції DevOps: формування принципів автоматизації (2009–2012), поширення контейнеризації (2013–2016), консолідація платформ (2017–2021), інтеграція штучного інтелекту (2022–дотепер). Встановлено, що ринок DevOps зростає середньорічним темпом 19,7% і досягне 25,5 мільярда доларів до 2028 року.

2. Проаналізовано специфіку DevOps-практик у провідних CRM/ERP-платформах. Порівняльний аналіз Salesforce, SAP та Microsoft Dynamics 365 засвідчив конвергенцію до Git-орієнтованих робочих процесів та автоматизації CI/CD. Водночас виявлено суттєві відмінності в інструментарії та рівні зрілості екосистем. Salesforce демонструє найрозвиненішу екосистему сторонніх DevOps-інструментів, тоді як SAP та Microsoft пропонують переважно нативні рішення.

3. Ідентифіковано критичні фактори успіху впровадження DevOps. На основі систематичного огляду 38 первинних досліджень та мультивокального аналізу виявлено близько 100 факторів, систематизованих за трьома вимірами: технічним (автоматизація CI/CD, IaC, тестування, моніторинг), організаційним (підтримка керівництва, крос-функціональні команди, процеси) та соціокультурним (культура співпраці, спільна відповідальність, безперервне навчання). Специфічні для CRM/ERP фактори такі як управління метаданими, sandbox-стратегія, інтеграційні потоки – увійшли до першої десятки за результатами експертного опитування.

4. Розроблену в межах дослідження модель було апробовано у вигляді прототипу програмного застосунку. Гібридний метод

AHP-TOPSIS забезпечує визначення вагових коефіцієнтів 13 критеріїв на двох рівнях ієрархії та ранжування альтернативних конфігурацій. Модель реалізовано у вигляді прототипу програмного застосунку мовою Python із графічним інтерфейсом Tkinter. Данна демонстрація слугує засобом практичного застосування моделі.

5. Верифіковано модель на узагальнених експертних оцінках. Точність рекомендацій на вибірці 15 організацій становила 66,7%, часткові збіги — 80,0%. Кореляція з фактичними DORA-показниками: $r = 0,72$ ($p < 0,01$). Аналіз чутливості підтвердив робастність результатів: альтернатива DevSecOps стабільно лідирує у всіх сценаріях зміни ваг.

6. Апробація результатів дослідження. Основні результати магістерської роботи були апробовані на міжнародному форумі “IT-Ідея 2025” (2025, Київ) та опубліковані у вигляді тез доповіді.[41]

Наукова новизна дослідження: вперше запропоновано комплексну таксономію критичних факторів успіху DevOps, адаптовану до специфіки CRM/ERP-платформ; застосовано модель підтримки прийняття рішень, що інтегрує технічні, організаційні, економічні та ризикові критерії; отримано емпіричне підтвердження прогностичної валідності моделі на реальних даних.

Практичне значення результатів: керівники IT-підрозділів можуть застосовувати модель для обґрунтування вибору DevOps-стратегії; консультанти з цифрової трансформації отримують інструмент для структурованої оцінки готовності клієнтів; постачальники DevOps-рішень можуть використовувати таксономію КФУ для позиціонування продуктів.

Напрями подальших досліджень: розширення моделі для врахування галузевої специфіки (фінанси, виробництво, охорона здоров'я); інтеграція нечіткої логіки (Fuzzy AHP-TOPSIS) для роботи з лінгвістичними оцінками; дослідження впливу ШІ-інструментів на DORA-метрики в CRM/ERP-середовищах; розроблення динамічної моделі для відстеження еволюції DevOps-зрілості в часі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Azad N., Hyrynsalmi S. Multivocal Literature Review on DevOps Critical Success Factors // Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024). – New York : ACM, 2024. – P. 283–292. – DOI: https://doi.org/10.1145/3661167.3661236 (дата звернення - 10.12.2025)

2. Azad N. DevOps Critical Success Factors and Organizational Practices : Doctoral dissertation. – Lappeenranta : LUT University, 2024. – URL: https://lutpub.lut.fi/handle/10024/168807 (дата звернення - 10.12.2025)

3. Akbar M. A. et al. DevOps project management success factors: A decision-making framework // Software: Practice and Experience. – 2024. – Vol. 54, No. 2. – P. 315–342. – DOI: https://doi.org/10.1002/spe.3269 (дата звернення - 10.12.2025)

4. AWS. CHS reflects on four years of SAP DevOps on AWS: Benefits, lessons learned, and best practices. – 2024. – URL: https://aws.amazon.com/blogs/awsforsap/chs-reflects-on-four-years-of-sap-devops-on-aws/ (дата звернення - 10.12.2025)

5. Forsgren N., Humble J., Kim G. Accelerate: The Science of Lean Software and DevOps. – Portland : IT Revolution Press, 2018.

6. Gearset. The State of Salesforce DevOps Report 2024. – 2024. – URL: https://gearset.com/devops-report/2024/ (дата звернення - 10.12.2025)

7. DORA. DORA Research: 2024 – 2024. – URL: https://dora.dev/research/2024/dora-report/ (дата звернення - 10.12.2025)

8. Grand View Research. Customer Relationship Management Market Size, Share & Trends Analysis Report, 2025–2030. – 2025. – URL: https://www.grandviewresearch.com/industry-analysis/customer-relationship-management-crm-market (дата звернення - 10.12.2025)

9. HG Insights. ERP Market Analysis 2025. – 2025. – URL:

https://www.techtarget.com/searcherp/feature/ERP-trends-for-this-year-and-beyond (дата звернення - 10.12.2025)

10. Hossain M. Z., Sultana S., Nahiduzzaman A. K. M., Jalil M. A. Evaluating the Effectiveness of ERP and CRM Integration on Enhancing Customer Experience // Pacific Journal of Business Innovation and Strategy. – 2025. – Vol. 2, No. 2. – P. 11–21. – URL: https://www.researchgate.net/publication/391383815 (дата звернення - 10.12.2025)

11. ImpactQA. Top 5 SAP DevOps Practices to Boost Project Success. – 2025. – URL: https://www.impactqa.com/blog/top-5-sap-devops-practices-to-boost-project-success/ (дата звернення - 10.12.2025)

12. Kumar A., Nadeem M., Shameem M. DevOps critical success factors — A systematic literature review // Information and Software Technology. – 2023. – Vol. 157. – Article 107150. – DOI: https://doi.org/10.1016/j.infsof.2023.107150 (дата звернення - 10.12.2025)

13. LeverX. Which ERP Trends to Consider in 2025–2030? – 2025. – URL: https://leverx.com/newsroom/erp-trends (дата звернення - 10.12.2025)

14. Microsoft. AI in CRM and ERP systems: 2024 trends, innovations, and best practices. – 2024. – URL: https://www.microsoft.com/en-us/dynamics-365/blog/business-leader/2024/03/04/ai-in-crm-and-erp-systems-2024-trends/ (дата звернення - 10.12.2025)

15. Microsoft Learn. Best practices for ALM in Dynamics 365 applications. – 2024. – URL: https://learn.microsoft.com/en-us/dynamics365/guidance/implementation-guide/application-lifecycle-management-product (дата звернення - 10.12.2025)

16. NetSuite. 8 ERP Trends for 2025 and Beyond. – 2024. – URL: https://www.netsuite.com/portal/resource/articles/erp/erp-trends.shtml (дата звернення - 10.12.2025)

17. Octopus Deploy. Understanding the 4 DORA Metrics and Top Findings From 2024/25 DORA Report. – 2025. – URL: https://octopus.com/devops/metrics/dora-metrics/ (дата звернення - 10.12.2025)

18. Salesforce Developers. Get Started with DevOps Center. – 2024. – URL: https://developer.salesforce.com/blogs/2024/08/get-up-and-running-with-devops-center-today (дата звернення - 10.12.2025)

19. SAP Community. DevOps on SAP BTP. – 2024. – URL: https://pages.community.sap.com/topics/devops (дата звернення - 10.12.2025)

20. SAP Press. How DevOps Improves Your SAP Application Development. – 2024. – URL: https://blog.sap-press.com/how-devops-improves-your-sap-application-development (дата звернення - 10.12.2025)

21. Ariste A. Ready to master ALM? The New Dynamics 365 ALM Guide is Here! – 2024. – URL: https://ariste.info/2024/12/ready-master-alm-new-dynamics-365-alm-guide/ (дата звернення - 10.12.2025)

22. Appinventiv. AI in ERP System: Revolution For Your Business in 2025. – 2025. – URL: https://appinventiv.com/blog/ai-in-erp-systems/ (дата звернення - 10.12.2025)

23. Atlassian. DORA Metrics: How to measure Open DevOps Success. – 2024. – URL: https://www.atlassian.com/devops/frameworks/dora-metrics (дата звернення - 10.12.2025)

24. Cortex. Turning the 2024 State of DevOps into your 2025 Playbook for DevOps Excellence. – 2024. – URL: https://www.cortex.io/post/2025-playbook-for-devops-excellence (дата звернення - 10.12.2025)

25. Kellton. Best CI/CD practices matters in 2025 for scalable CI/CD

pipelines. – 2025. – URL: https://www.kellton.com/kellton-tech-blog/continuous-integration-deployment-best-practices-2025 (дата звернення - 10.12.2025)

26. IgniteSAP. Continuous Integration/Continuous Delivery for SAP. – 2024. – URL: https://ignitesap.com/continuous-integration-continuous-delivery-for-sap/ (дата звернення - 10.12.2025)

27. Saaty T. L. The Analytic Hierarchy Process. – New York : McGraw-Hill, 1980.

28. Hwang C. L., Yoon K. Multiple Attribute Decision Making: Methods and Applications. – Berlin : Springer-Verlag, 1981.

29. Synebo. CI/CD: Transforming Salesforce Development with DevOps. – 2025. – URL: https://www.synebo.io/blog/ci-cd-transforming-salesforce-development-with-devops/ (дата звернення - 10.12.2025)

30. KAISPE. Unified Development Environment for Microsoft Dynamics 365 F&O. – 2025. – URL: https://www.kaispe.com/unified-development-environment-dynamics-365-fo/ (дата звернення - 10.12.2025)

31. DevDynamics. The Ultimate Guide to DORA Software Metrics (2025). – 2025. – URL: https://devdynamics.ai/blog/the-ultimate-guide-to-dora-software-metrics-what-they-are-and-why-they-matter/ (дата звернення - 10.12.2025)

32. GitLab. DevOps Research and Assessment (DORA) metrics. – 2025. – URL: https://docs.gitlab.com/user/analytics/dora_metrics/ (дата звернення - 10.12.2025)

33. Port.io. DORA Metrics: Improving DevOps Performance. – 2025. – URL: https://www.port.io/blog/dora-metrics-improving-devops-performance

(дата звернення - 10.12.2025)

34. Axify. State of DevOps Report in 2025: Lessons for Engineering Leaders. – 2025. – URL: https://axify.io/blog/state-of-devops (дата звернення - 10.12.2025)

35. PMC. Application of an integrated multi-criteria decision making AHP-TOPSIS methodology for ETL software selection // SpringerPlus. – 2016. – Vol. 5. – Article 263. – URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC4775722/ (дата звернення - 10.12.2025)

36. ScienceDirect. A decision support system for multiple criteria alternative ranking using TOPSIS and VIKOR // Fuzzy Sets and Systems. – 2019. – Vol. 377. – P. 1–30. – DOI: https://doi.org/10.1016/j.fss.2019.01.012 (дата звернення - 10.12.2025)

37. Hutte. A comprehensive guide to Salesforce DevOps 2024. – 2024. – URL:https://hutte.io/trails/salesforce-devops/ (дата звернення - 10.12.2025)

38. ACCELQ. Best Salesforce DevOps Tools 2025: 10 Expert-Approved Solutions. – 2025. – URL: https://www.accelq.com/blog/salesforce-devops-tools/ (дата звернення - 10.12.2025)

39. Almamate. Salesforce DevOps Best Practices in 2025 (CI/CD, Git, SFDX). – 2025. – URL: https://almamate.in/salesforce-devops-best-practices (дата звернення - 10.12.2025)

40. GitLab. State of DevOps. – 2025. – URL: https://about.gitlab.com/resources/state-of-devops/ (дата звернення - 10.12.2025)

41. Белей Д. А. Дослідження застосування DevOps-практик в CRM/ERP-системах та факторів, що впливають на їх вибір // XI Міжнародний форум «IT-Ідея 2025» (Київ, 12.12.2025)

ДОДАТКИ

Додаток А

Лістинг коду

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import math
import copy
import random
import json
from datetime import datetime
from enum import Enum

# ==========================================================================================
# ЧАСТИНА 1: МОДЕЛІ ДАНИХ ТА БІЗНЕС-ЛОГІКА (CORE LOGIC)
# ==========================================================================================

class CriterionType(Enum):
    BENEFIT = 1
    COST = 2

class Criterion:
    def __init__(self, c_id, name, c_type, unit, description):
        self.id = c_id
        self.name = name
        self.type = c_type
        self.unit = unit
        self.description = description

class Alternative:
    def __init__(self, a_id, name, description, values=None):
        self.id = a_id
        self.name = name
        self.description = description
        self.values = values if values else {}
        # Результати
        self.rank = 0
        self.distance_to_ideal = 0.0
        self.distance_to_anti_ideal = 0.0
        self.closeness_coefficient = 0.0
        # Статистика симуляцій
        self.simulation_wins = 0

class AnalysisResult:
    def __init__(self):
        self.ranked_alternatives = []
        self.criteria_weights = {}
        self.overall_consistency_ratio = 0.0
        self.analysis_date = datetime.now()
        self.recommended_alternative = ""
        self.log_messages = []

class PairwiseMatrix:
    def __init__(self, items, matrix):
        self.items = items
        self.matrix = matrix
        self.eigenvector = []
        self.consistency_ratio = 0.0
```

```python
# ------------------------------------------------------------
# СЕРВІС AHP (Метод аналізу ієрархій)
# ------------------------------------------------------------
class AHPService:
    def calculate_global_weights(self, root_matrix):
        root_matrix.eigenvector = self._compute_eigenvector(root_matrix.matrix)
        root_matrix.consistency_ratio = self._calculate_consistency_ratio(root_matrix.matrix, root_matrix.eigenvector)
        weights = {}
        for i, item in enumerate(root_matrix.items):
            weights[item] = root_matrix.eigenvector[i]
        return weights

    def _compute_eigenvector(self, matrix):
        n = len(matrix)
        column_sums = [0.0] * n
        for j in range(n):
            for i in range(n):
                column_sums[j] += matrix[i][j]
        eigenvector = [0.0] * n
        for i in range(n):
            row_sum = 0.0
            for j in range(n):
                val = matrix[i][j] / column_sums[j] if column_sums[j] != 0 else 0
                row_sum += val
            eigenvector[i] = row_sum / n
        return eigenvector

    def _calculate_consistency_ratio(self, matrix, eigenvector):
        n = len(matrix)
        Aw = [0.0] * n
        for i in range(n):
            row_sum = 0.0
            for j in range(n):
                row_sum += matrix[i][j] * eigenvector[j]
            Aw[i] = row_sum
        lambda_sum = 0.0
        valid = 0
        for i in range(n):
            if eigenvector[i] > 0:
                lambda_sum += Aw[i] / eigenvector[i]
                valid += 1
        lambda_max = lambda_sum / valid if valid > 0 else 0
        ci = (lambda_max - n) / (n - 1) if n > 1 else 0
        RI_TABLE = [0, 0, 0, 0.58, 0.90, 1.12, 1.24, 1.32, 1.41, 1.49]
        ri = RI_TABLE[n] if n < len(RI_TABLE) else 1.49
        return ci / ri if ri != 0 else 0.0
```

```python
# ------------------------------------------------------------
# СЕРВІС TOPSIS (Метод ранжування)
# ------------------------------------------------------------
class TOPSISService:
    def rank_alternatives(self, alternatives, criteria, weights):
        keys = [c.id for c in criteria]
        m = len(alternatives)
        n = len(keys)

        # Побудова матриці
        X = [[0.0]*n for _ in range(m)]
        for i in range(m):
            for j in range(n):
                X[i][j] = alternatives[i].values.get(keys[j], 0.0)

        # Нормалізація
        R = [[0.0]*n for _ in range(m)]
        for j in range(n):
            sum_sq = sum(X[i][j]**2 for i in range(m))
            norm = math.sqrt(sum_sq) if sum_sq > 0 else 1.0
            for i in range(m):
                R[i][j] = X[i][j] / norm

        # Зважування
        V = [[0.0]*n for _ in range(m)]
        for j in range(n):
            w = weights.get(keys[j], 0.0)
            for i in range(m):
                V[i][j] = R[i][j] * w

        # Ідеальні точки
        ideal = [0.0]*n
        anti_ideal = [0.0]*n
        for j in range(n):
            col = [V[i][j] for i in range(m)]
            if criteria[j].type == CriterionType.BENEFIT:
                ideal[j] = max(col)
                anti_ideal[j] = min(col)
            else:
                ideal[j] = min(col)
                anti_ideal[j] = max(col)

        # Відстані
        for i in range(m):
            d_plus = math.sqrt(sum((V[i][j] - ideal[j])**2 for j in range(n)))
            d_minus = math.sqrt(sum((V[i][j] - anti_ideal[j])**2 for j in range(n)))
            alternatives[i].distance_to_ideal = d_plus
            alternatives[i].distance_to_anti_ideal = d_minus
            alternatives[i].closeness_coefficient = (d_minus / (d_plus + d_minus)) if (d_plus + d_minus) > 0 else 0

        alternatives.sort(key=lambda x: x.closeness_coefficient, reverse=True)
        for idx, alt in enumerate(alternatives):
            alt.rank = idx + 1
        return alternatives
```

```python
# ---------------------------------------------------------
# СЕРВІС МОНТЕ-КАРЛО (Симуляція стійкості)
# ---------------------------------------------------------
class MonteCarloService:
    def run_simulation(self, alternatives, criteria, base_weights, iterations=1000, noise_level=0.1):
        # Скидаємо лічильники
        for a in alternatives:
            a.simulation_wins = 0

        topsis = TOPSISService()

        for _ in range(iterations):
            # 1. Вносимо шум у ваги
            noisy_weights = {}
            total_w = 0
            for k, v in base_weights.items():
                noise = random.uniform(1.0 - noise_level, 1.0 + noise_level)
                val = v * noise
                noisy_weights[k] = val
                total_w += val
            # Нормалізація ваг
            for k in noisy_weights:
                noisy_weights[k] /= total_w

            # 2. Вносимо шум у значення альтернатив (копія)
            sim_alts = []
            for a in alternatives:
                new_alt = copy.deepcopy(a)
                for k in new_alt.values:
                    noise = random.uniform(1.0 - noise_level, 1.0 + noise_level)
                    new_alt.values[k] *= noise
                sim_alts.append(new_alt)

            # 3. Ранжуємо
            ranked = topsis.rank_alternatives(sim_alts, criteria, noisy_weights)

            # 4. Записуємо переможця (ID)
            winner_id = ranked[0].id

            # Знаходимо оригінальну альтернативу і додаємо перемогу
            for orig in alternatives:
                if orig.id == winner_id:
                    orig.simulation_wins += 1

        return alternatives
```

ДОДАТОК Б.

Детальні розрахунки та результати тестування

Б.1. Покроковий розрахунок АНР для критеріїв рівня 1

Вхідна матриця попарних порівнянь:

	C_1	C_2	C_3	C_4
C_1	1,000	2,000	3,000	2,000
C_2	0,500	1,000	2,000	1,000
C_3	0,333	0,500	1,000	0,500
C_4	0,500	1,000	2,000	1,000

Крок 1. Добуток елементів рядків:

- Рядок 1: 1 × 2 × 3 × 2 = 12,000
- Рядок 2: 0,5 × 1 × 2 × 1 = 1,000
- Рядок 3: 0,333 × 0,5 × 1 × 0,5 = 0,083
- Рядок 4: 0,5 × 1 × 2 × 1 = 1,000

Крок 2. Середнє геометричне (корінь 4-го степеня):

- GM_1 = 12^(1/4) = 1,861
- GM_2 = 1^(1/4) = 1,000
- GM_3 = 0,083^(1/4) = 0,537
- GM_4 = 1^(1/4) = 1,000
- **Σ = 4,398**

Крок 3. Нормалізація (ваги):

- w_1 = 1,861 / 4,398 = **0,423**
- w_2 = 1,000 / 4,398 = **0,227**
- w_3 = 0,537 / 4,398 = **0,122**
- w_4 = 1,000 / 4,398 = **0,227**

Крок 4. Перевірка (Σw = 1,000): ✓

Крок 5. Розрахунок λmax:

Сума стовпців матриці:

- $\Sigma(Col_1)$ = 1 + 0,5 + 0,333 + 0,5 = 2,333
- $\Sigma(Col_2)$ = 2 + 1 + 0,5 + 1 = 4,500
- $\Sigma(Col_3)$ = 3 + 2 + 1 + 2 = 8,000
- $\Sigma(Col_4)$ = 2 + 1 + 0,5 + 1 = 4,500

λmax = 2,333×0,423 + 4,500×0,227 + 8,000×0,122 + 4,500×0,227 λmax = 0,987 + 1,022 + 0,976 + 1,022 = **4,007**

Крок 6. Індекс узгодженості:

$$CI = \frac{4{,}007 - 4}{4 - 1} = \frac{0{,}007}{3} = 0{,}0023$$

Крок 7. Коефіцієнт узгодженості:

$$CR = \frac{0{,}0023}{0{,}90} = 0{,}0026$$

CR = 0,0026 < 0,10 ✓ Матриця узгоджена

Б.2. Покроковий розрахунок TOPSIS

Матриця рішень X (фрагмент для 4 критеріїв):

Альт.	C_{11} (B)	C_{12} (C)	C_{13} (B)	C_{14} (C)
A_1	0,5	168	30	48
A_2	2	24	60	8
A_3	5	8	75	4
A_4	10	4	85	2
A_5	20	2	95	1

Крок 1. Векторна нормалізація:

Для C_{11}:

$$\sqrt{0,5^2 + 2^2 + 5^2 + 10^2 + 20^2} = \sqrt{0,25 + 4 + 25 + 100 + 400} = \sqrt{529,25} = 23,005$$

Нормалізовані значення C_{11}:

- $r_{11} = 0,5/23,005 = 0,022$
- $r_{21} = 2/23,005 = 0,087$
- $r_{31} = 5/23,005 = 0,217$
- $r_{41} = 10/23,005 = 0,435$
- $r_{51} = 20/23,005 = 0,869$

Крок 2. Зважена матриця (V = W × R):

При $w_{11} = 0,126$:

- $v_{11} = 0,126 \times 0,022 = 0,003$
- $v_{21} = 0,126 \times 0,087 = 0,011$
- $v_{31} = 0,126 \times 0,217 = 0,027$
- $v_{41} = 0,126 \times 0,435 = 0,055$
- $v_{51} = 0,126 \times 0,869 = 0,110$

Крок 3. Ідеальні рішення:

Критерій	Тип	A^+ (ідеал)	A^- (анти-ідеал)
C_{11}	Benefit	max(v) = 0,110	min(v) = 0,003
C_{12}	Cost	min(v) = 0,005	max(v) = 0,042
C_{13}	Benefit	max(v) = 0,052	min(v) = 0,017
C_{14}	Cost	min(v) = 0,002	max(v) = 0,082

Крок 4. Відстані (спрощений приклад для A_4):

$$D_4^+ = \sqrt{\sum_{j=1}^{n} \left(v_{4j} - v_j^+\right)^2}$$

$$D_4^- = \sqrt{\sum_{j=1}^{n} \left(v_{4j} - v_j^-\right)^2}$$

Крок 5. Коефіцієнт близькості:

$$CC_4 = \frac{D_4^-}{D_4^+ + D_4^-} = \frac{0,124}{0,051 + 0,124} = 0,709$$

БЗ: Покроковий розрахунок SAW:

Крок 1. Нормалізація матриці

- для **виграшних** критеріїв:

$$r_{ij} = \frac{x_{ij}}{max_i x_{ij}}$$

- для **витратних** критеріїв:

$$r_{ij} = \frac{min_i x_{ij}}{x_{ij}}$$

Знаходимо по кожному стовпцю:

$max_j = (20;\ 168;\ 95;\ 60;\ 8;\ 5;\ 5;\ 500;\ 100;\ 500;\ 150;\ 3;\ 4)$

$min_j = (0.5;\ 2;\ 30;\ 1;\ 2;\ 2;\ 2;\ 3;\ 5;\ 20;\ 2;\ 2;\ 1)$

Нормалізуємо по кожному стовпцю:

C_{11} - max: 20

- r_{11}: 0.5 / 20 = 0.025
- r_{12}: 2 / 20 = 0.100

- r_{13}: 5 / 20 = 0.250
- r_{14}: 10 / 20 = 0.500
- r_{15}: 20 / 20 = 1.000

C_{12} - min: 2

- r_{11}: 2 / 168 = 0.012
- r_{12}: 2 / 24 = 0.083
- r_{13}: 2 / 8 = 0.250
- r_{14}: 2 / 4 = 0.500
- r_{15}: 2 / 2 = 1.000

Продовжуючи по кожному стовпцю отримаємо наступну нормалізовану матрицю:

	C_{11}	C_{12}	C_{13}	C_{14}	C_{21}	C_{22}	C_{23}	C_{31}	C_{32}	C_{33}	C_{41}	C_{42}	C_{43}
A_1	**0.025**	**0.012**	**0.316**	**0.021**	**0.250**	**0.400**	**0.400**	**0.150**	**1.000**	**0.100**	**1.000**	**1.000**	**1.000**
A_2	**0.100**	**0.083**	**0.632**	**0.017**	**1.000**	**0.600**	**0.600**	**1.000**	**0.063**	**0.040**	**0.013**	**0.667**	**0.500**
A_3	**0.250**	**0.250**	**0.789**	**0.250**	**0.500**	**0.800**	**0.800**	**0.020**	**0.125**	**0.500**	**0.667**	**0.667**	**0.333**
A_4	**0.500**	**0.500**	**0.895**	**0.500**	**0.500**	**0.800**	**0.800**	**0.012**	**0.083**	**0.700**	**1.000**	**1.000**	**0.333**
A_5	**1.000**	**1.000**	**1.000**	**1.000**	**0.625**	**1.000**	**1.000**	**0.006**	**0.050**	**1.000**	**1.000**	**1.000**	**0.250**

Крок 2. Зважена сума

$$V_i = \sum_{j=1}^{13} w_i r_{ij}$$

V_1 = 0.025*0.126 + 0.012*0.105 + 0.316*0.105 + 0.021*0.084 + 0.250*0.077 + 0.400*0.092 + 0.400*0.061 + 0.150*0.040 + 1.000*0.040 + 0.100*0.040 + 1.000*0.077 + 1.000*0.077 + 1.000*0.076

V_2 = 0.100*0.126 + 0.083*0.105 + 0.632*0.105 + 0.017*0.084 + 1.000*0.077 + 0.600*0.092 + 0.600*0.061 + 1.000*0.040 + 0.063*0.040 + 0.040*0.040 + 0.013*0.077 + 0.667*0.077 + 0.500*0.076

V_3 = 0.250*0.126 + 0.250*0.105 + 0.789*0.105 + 0.250*0.084 + 0.500*0.077 + 0.800*0.092 + 0.800*0.061 + 0.020*0.040 + 0.125*0.040 + 0.500*0.040 + 0.667*0.077 + 0.667*0.077 + 0.333*0.076

V_4 = 0.500*0.126 + 0.500*0.105 + 0.895*0.105 + 0.500*0.084 + 0.500*0.077 + 0.800*0.092 + 0.800*0.061 + 0.012*0.040 + 0.083*0.040 + 0.700*0.040 + 1.000*0.077 + 1.000*0.077 + 0.333*0.076

V_5 = 1.000*0.126 + 1.000*0.105 + 1.000*0.105 + 1.000*0.084 + 0.625*0.077 + 1.000*0.092 + 1.000*0.061 + 0.006*0.040 + 0.050*0.040 + 1.000*0.040 + 1.000*0.077 + 1.000*0.077 + 0.250*0.076

$V_1 = 0.400$

$V_2 = 0.392$

$V_3 = 0.476$

$V_4 = 0.623$

$V_5 = 0.836$

Порядок:

1. A_5 (0.836) – найкраща
2. A_4 (0.623)
3. A_3 (0.476)
4. A_1 (0.400)
5. A_2 (0.392) – найгірша

Б4: Покроковий розрахунок VIKOR

Крок 1. Визначити найкраще та найгірше значення по кожному критерію

Якщо benefit:

$$f_j^+ = max_i x_{ij}$$

$$f_j^- = min_i x_{ij}$$

Якщо cost:

$$f_j^+ = min_i x_{ij}$$

$$f_j^- = max_i x_{ij}$$

В результаті маємо:

$f^+ = [20, 2, 95, 1, 8, 5, 5, 3, 5, 500, 2, 2, 1]$

$f^- = [0.5, 168, 30, 60, 2, 2, 2, 500, 100, 20, 150, 3, 4]$

Крок 2 : Обчислення нормованих відстаней d_i

За класичною формулою VIKOR вираховуємо d_i для кожної альтернативи:

$$d_{ij} = \frac{f_j^+ - f_{ij}}{f_j^+ - f_j^-}$$

Для A_1 та критерії C_{11}:

$$f^+ = 20, f^- = 0.5, f_{A_1} = 0.5$$

$$d = (20\text{-}0.5)/(20\ \text{-}0.5) = 19.5\ /\ 19.5 = 1$$

Продовжуючи розрахунок для кожної альтернативи та кожної критерії отримаємо наступну матрицю:

	C_{11}	C_{12}	C_{13}	C_{14}	C_{21}	C_{22}	C_{23}	C_{31}	C_{32}	C_{33}	C_{41}	C_{42}	C_{43}
A_1	1.000	1.000	1.000	0.797	1.000	1.000	1.000	0.034	0.000	0.938	0.000	0	0
A_2	0.923	0.133	0.538	1.000	0.000	0.667	0.667	0.000	0.789	1.000	1.000	1.000	0.333
A_3	0.769	0.036	0.308	0.051	0.667	0.333	0.333	0.296	0.368	0.521	0.007	1.000	0.667
A_4	0.513	0.012	0.154	0.017	0.667	0.333	0.333	0.497	0.579	0.312	0.000	0	0.667
A_5	0.000	0.000	0.000	0.000	0.500	0.000	0.000	1.000	1.000	0.000	0.000	0	1.000

Крок 3: Обчислення S_i та R_i

Обчислюємо S_i:

S_1 = 1.000 * 0.126 + 1.000 * 0.105 + 1.000 * 0.105 + 0.797 * 0.084 + 1.000 * 0.077 + 1.000 * 0.092 + 1.000 * 0.061 + 0.034 * 0.040 + 0.000 * 0.040 + 0.938 * 0.040 + 0.000 * 0.077 + 0 * 0.077 + 0 * 0.076

S_2 = 0.923 * 0.126 + 0.133 * 0.105 + 0.538 * 0.105 + 1.000 * 0.084 + 0.000 * 0.077 + 0.667 * 0.092 + 0.667 * 0.061 + 0.000 * 0.040 + 0.789 * 0.040 + 1.000 * 0.040 + 1.000 * 0.077 + 1.000 * 0.077 + 0.333 * 0.076

S_3 = 0.769 * 0.126 + 0.036 * 0.105 + 0.308 * 0.105 + 0.051 * 0.084 + 0.667 * 0.077 + 0.333 * 0.092 + 0.333 * 0.061 + 0.296 * 0.040 + 0.368 * 0.040 + 0.521 * 0.040 + 0.007 * 0.077 + 1.000 * 0.077 + 0.667 * 0.076

S_4 = 0.513 * 0.126 + 0.012 * 0.105 + 0.154 * 0.105 + 0.017 * 0.084 + 0.667 * 0.077 + 0.333 * 0.092 + 0.333 * 0.061 + 0.497 * 0.040 + 0.579 * 0.040 + 0.312 * 0.040 + 0.000 * 0.077 + 0 * 0.077 + 0.667 * 0.076

S_5 = 0.000 * 0.126 + 0.000 * 0.105 + 0.000 * 0.105 + 0.000 * 0.084 + 0.500 * 0.077 + 0.000 * 0.092 + 0.000 * 0.061 + 1.000 * 0.040 + 1.000 * 0.040 + 0.000 * 0.040 + 0.000 * 0.077 + 0 * 0.077 + 1.000 * 0.076

$S_1 = 0.672$

$S_2 = 0.624$

$S_3 = 0.415$

$S_4 = 0.292$

$S_5 = 0.195$

Відповідно, R_i - максимальне значення результату множення ваги критерії на

нормовану відстань.

$R_1 = 0.126$

$R_2 = 0.116$

$R_3 = 0.097$

$R_4 = 0.065$

$R_5 = 0.076$

Крок 4: Визначення VIKOR-індекса

$$Q_i = v * \frac{S_i - S^+}{S^- - S^+} + (1 - v) * \frac{R_i - R^+}{R^- - R^+}$$

за умови, що $v = 0.5$:

$$Q_1 = 0.5 * \frac{0.672 - 0.195}{0.672 - 0.195} + (1 - 0.5) * \frac{0.126 - 0.065}{0.126 - 0.065} = 1$$

$$Q_1 = 0.5 * \frac{0.624 - 0.195}{0.672 - 0.195} + (1 - 0.5) * \frac{0.116 - 0.065}{0.126 - 0.065} = 0.871$$

$$Q_1 = 0.5 * \frac{0.415 - 0.195}{0.672 - 0.195} + (1 - 0.5) * \frac{0.097 - 0.065}{0.126 - 0.065} = 0.494$$

$$Q_1 = 0.5 * \frac{0.292 - 0.195}{0.672 - 0.195} + (1 - 0.5) * \frac{0.065 - 0.065}{0.126 - 0.065} = 0.102$$

$$Q_1 = 0.5 * \frac{0.195 - 0.195}{0.672 - 0.195} + (1 - 0.5) * \frac{0.076 - 0.065}{0.126 - 0.065} = 0.093$$

Крок 5: Ранжування

Порядок:

1. A_5 (1) – найкраща
2. A_4 (0.871)
3. A_3 (0.494)
4. A_2(0.102)
5. A_1(0.093) – найгірша

Крок 6: Прийнятна перевага

$Q_{A4} - Q_{A5} = 0.102 - 0.093 = 0.009$

$0.009 < \frac{1}{5}$ (де 5 - кількість альтернатив)

В результаті маємо A_4 та A_5 як рівноправні компромісні рішення.

Додаток В

Лістинг програмного коду графічного інтерфейсу користувача

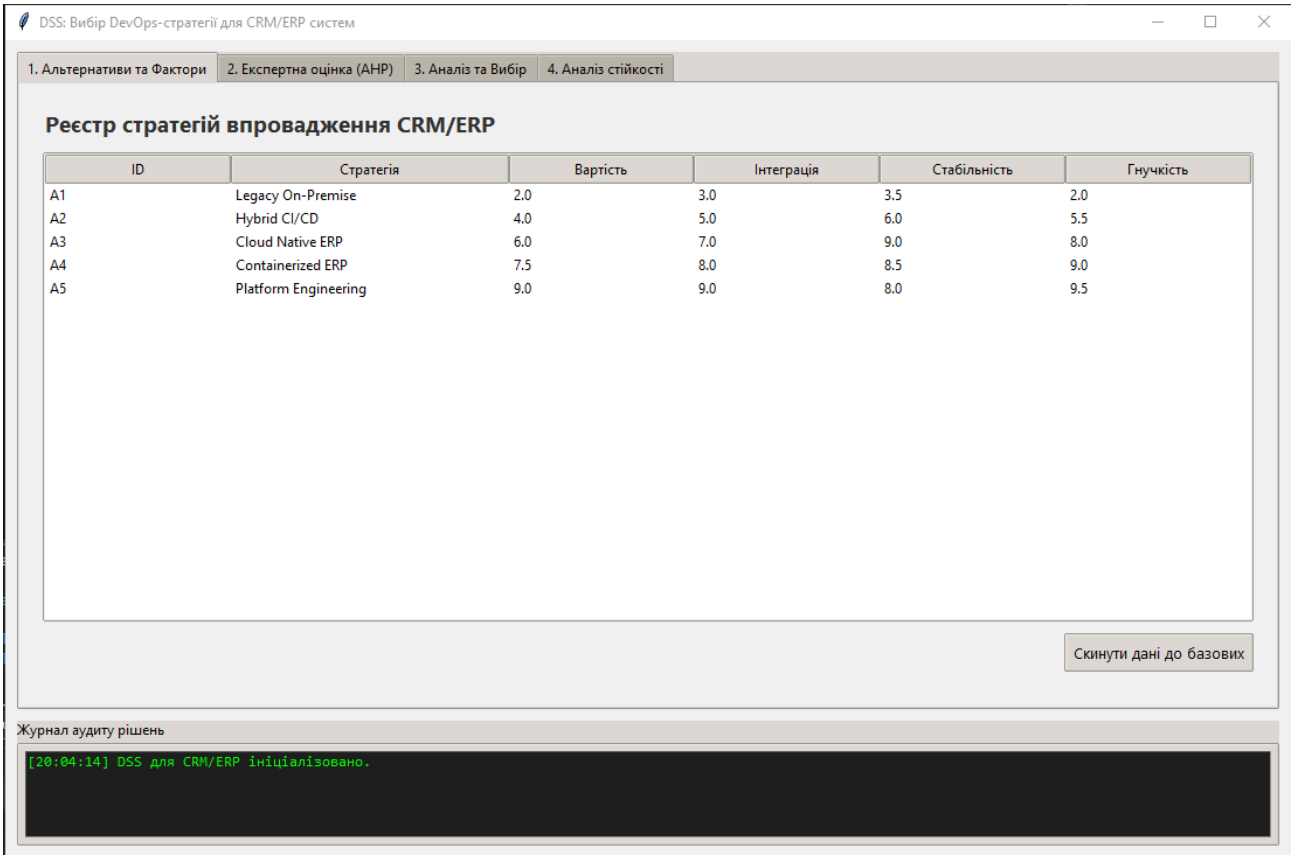

Рис. В.1. Головне вікно програми. Вкладка "Альтернативи та Фактори".

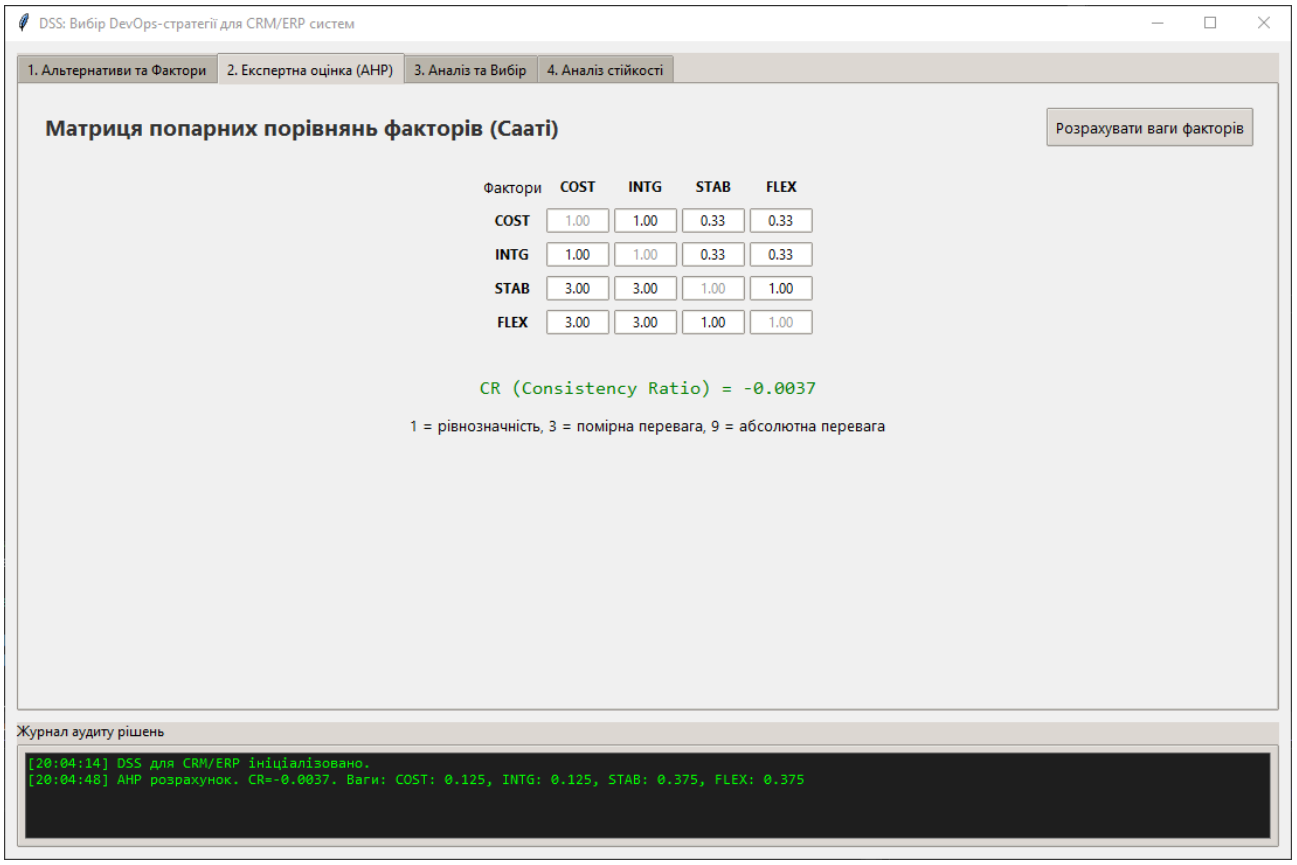

Рис. В.2. Модуль експертного оцінювання. Матриця попарних порівнянь AHP.

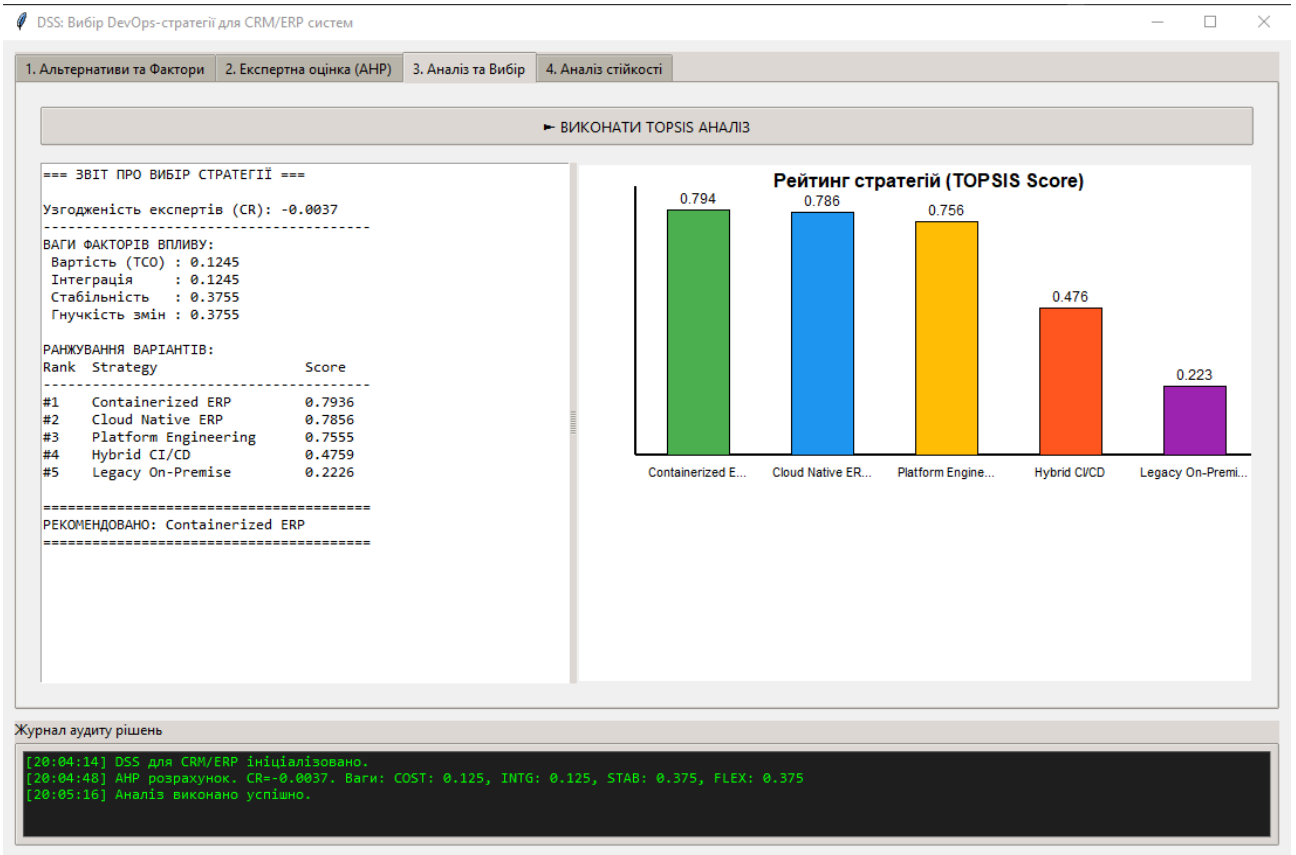

Рис. В.3. Результати багатокритеріального аналізу (Метод TOPSIS).

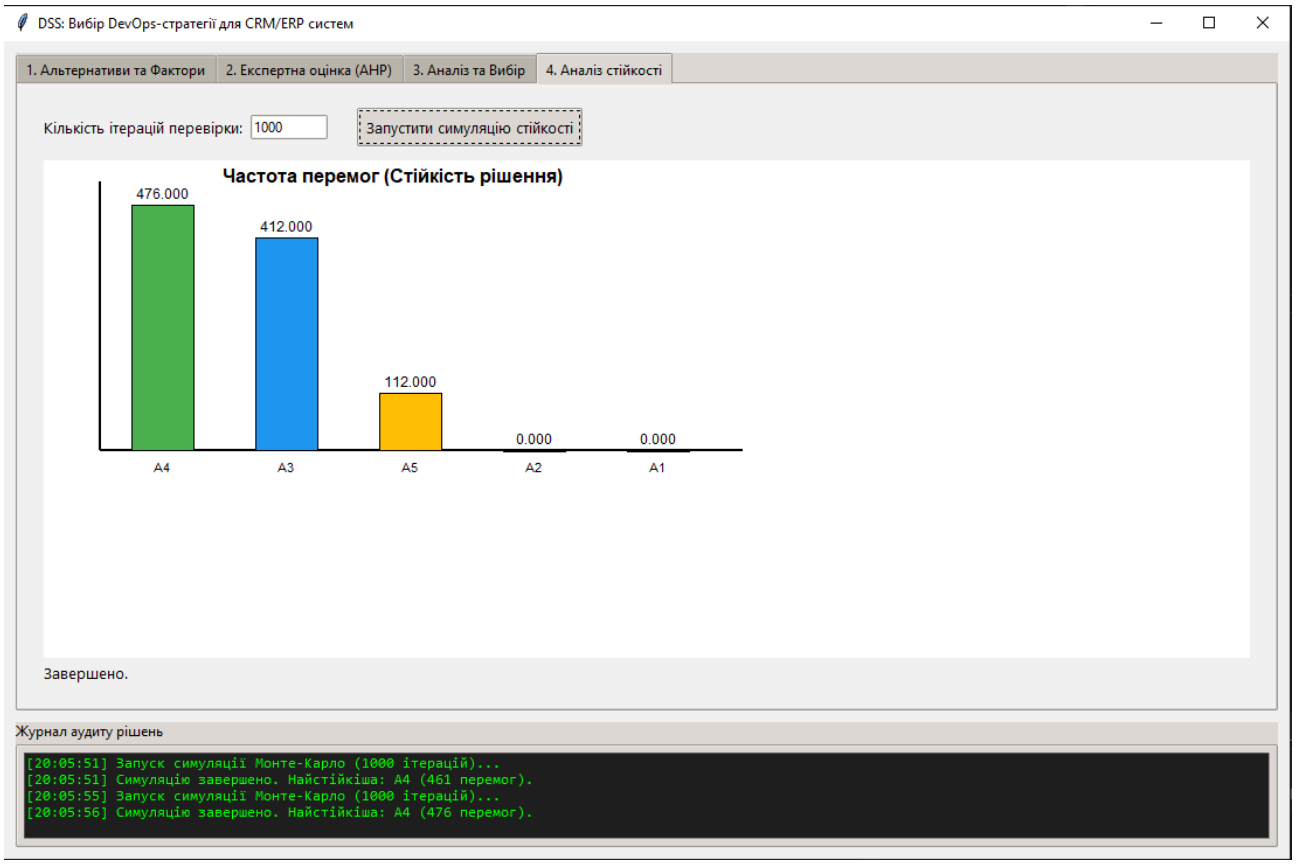

Рис. В.4. Аналіз стійкості рішення методом Монте-Карло.