КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 «Комп'ютерні науки»
Освітньо-наукова програма «Управління проєктами»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА
на тему:
"Дослідження процесів управління проєктами розробки інформаційної системи для підтримки прийняття управлінських рішень в ІТ-компаніях"

Студента 2-го курсу групи УП-21

Максима ЕЛЛІСЕНА

Науковий керівник:

кандидат техн. наук, професор
Віктор МОРОЗОВ

_____ _____ _____
(підпис студента) *(дата)* *(підпис)*

Попередній захист:

_____
(Висновок: "До захисту в Екзаменаційній комісії")

Завідувач кафедри
технологій управління _____ Морозов В.В _____
(підпис) *(прізвище, ініціали)* *(дата)*

Київ – 2026

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління

Освітній рівень: Магістр

Спеціальність: 122 Комп'ютерні науки

Освітня програма: Управління проєктами

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Морозов В.В.

_____
"8" грудня 2025 року

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студента Еллісена Максима Андрійовича

Група УП-21

1. Тема кваліфікаційної роботи "Дослідження процесів управління проєктами розробки інформаційної системи для підтримки прийняття управлінських рішень в IT-компаніях"

Затверджена на засіданні кафедри технологій управління, протокол № 4 від "5" грудня 2025 р.

2. Строк подання студентом готової роботи – "15" травня 2026 р.

3. Цільова установка та вихідні дані до роботи: дослідження проблематики управління рішеннями в IT-компаніях, планування розробки інформаційної системи Decision Intelligence «DECY» (календарне планування, кошторис, тощо), математичне моделювання процесів пов'язаних з управлінням рішеннями в рамках управління проєктами (FSM для життєвого циклу рішення, DAG для графу залежностей), проєктування IT-архітектури та планування пілотного впровадження.

4. Зміст роботи: аналіз проблеми неформалізованості процесів управління рішеннями в сучасних IT-компаніях, маркетингове дослідження B2B-ринку Decision Intelligence, стратегічний аналіз проєкту (дерево проблем і цілей, SWOT-аналіз, модель Портера), розробка моделі життєвого циклу рішення (FSM) та моделі взаємозв'язків між рішеннями (DAG), побудова ієрархічної структури робіт (WBS), ресурсно-вартісне планування та розробка системи контролю на основі EVM, проєктування концептуальної архітектури, логічної моделі бази даних, розробка інтерфейсів, оцінка ризиків проєкту, планування впровадження, висновки.

5. Перелік графічного матеріалу (слайдів): титульна сторінка, актуальність та мета роботи, дерево проблем та цілей проєкту, SWOT-аналіз та конкурентне позиціонування платформи «DECY», математична модель життєвого циклу рішення (діаграма станів FSM) та функція спадання актуальності, граф залежностей рішень (DAG) та каскадний аналіз, ієрархічна структура робіт (WBS) та календарний план (діаграма Ганта), архітектура системи (C4), ER-діаграма бази даних, користувацькі інтерфейси платформи, пілотне впровадження проєкту, висновки.

6. Календарний план виконання роботи:

№ з/п	**Назва частини роботи**	**План виконання роботи**
1	Вибір теми кваліфікаційної роботи магістра (КРМ)	До 15.11.25
2	Підготовка вступу	До 26.12.25
3	Підготовка розділу 1	08.01.26 – 18.02.26
4	Підготовка розділу 2	18.02.26 – 30.03.26
5	Підготовка розділу 3	01.04.26 – 20.04.26
6	Підготовка розділу 4	21.04.26 – 05.05.26
7	Остаточне оформлення кваліфікаційної роботи	05.05.26 – 10.05.26
8	Передача КРМ в електронному вигляді на кафедру на перевірку роботи на плагіат	10.05.26 – 12.05.26
9	Передача КРМ (друк.) на рецензію керівнику	До 15.05.26

№ з/п	Назва частини роботи	План виконання роботи
10	Презентація кваліфікаційної роботи магістра. Попередній захист роботи на кафедрі	15.05.26 – 20.05.26
11	Передача КРМ (друк.) на рецензію	21.05.26
12	Передача роздрукованої та переплетеної роботи на кафедру	22.05.26
13	Захист магістерської кваліфікаційної роботи	25.05.26 – 26.05.26

Дата видачі завдання: “9” грудня 2025 р.

Керівник роботи: кандидат техн. наук, професор Морозов В. В.

_____

(підпис)

Завдання прийняв до виконання:

студент групи УП-21 Еллісен М.А.

_____

(підпис)

ЗМІСТ

АНОТАЦІЯ

кваліфікаційної роботи магістра на тему

“Дослідження процесів управління проєктами розробки інформаційної системи для підтримки прийняття управлінських рішень в IT-компаніях”

Студент: Еллісен Максим Андрійович

Науковий керівник: Морозов Віктор Володимирович

Рік захисту – 2026

Тема кваліфікаційної роботи – “Дослідження процесів управління проєктами розробки інформаційної системи для підтримки прийняття управлінських рішень в IT-компаніях”. Метою дослідження є планування та розробка ефективних процесів управління проєктом з розробки інформаційної платформи Decision Intelligence, а також дослідження та вирішення проблем пов’язаних з управлінням рішеннями в розрізі управління проєктами. Об’єкт дослідження – розробка інформаційної системи зі штучним інтелектом для менеджменту та контролю управлінських рішень. Предметом дослідження є процеси управління проєктами з розробки програмного забезпечення, які впливають на ефективність створення платформи «DECY» – управління вартістю, часом, ризиками, а також вирішення проблематики менеджменту управлінських рішень у IT-проєктах.

Особливістю даної роботи є її подвійний фокус у сфері управління проєктами. З одного боку, досліджуються та застосовуються класичні процеси управління IT-проєктом для створення програмного забезпечення (планування розкладу, формування бюджету, оцінка ризиків). З іншого боку, сам продукт розробки – система «DECY» – виступає спеціалізованим інструментом проєктного менеджменту. Для вирішення суттєвих управлінських проблем, пов’язаних з управлінням рішеннями (борг рішень, втрата контексту, інституційна амнезія), в основу платформи закладено спеціалізовані

математичні моделі, які спрямовані на вирішення проблематики управління самими рішеннями у дисципліні управління проєктами.

Робота містить аналіз сучасних підходів до управління IT-проєктами та оцінку ринкового потенціалу B2B SaaS-продуктів у досліджуваній ніші. На цій базі сформовано план розробки інформаційної системи: побудовано ієрархічну структуру робіт (WBS), розраховано календарний графік та складено кошторис. Окрему увагу приділено концептуальній IT-архітектурі, проєктуванню бази даних, управлінню проєктними загрозами та підготовці до пілотного впровадження.

У першому розділі здійснено огляд предметної області, зосереджений на проблемі менеджменту управлінських рішень у IT-компаніях. Сформульовано цілі, проведено маркетингове та конкурентне дослідження (SWOT-аналіз), а також обґрунтовано стратегічне позиціонування інформаційної системи як продукту.

Другий розділ зосереджений на розробці математичних моделей для вирішення проблематики управління рішеннями, декомпозиції робіт проєкту (WBS) та ресурсно-вартісному плануванні проєкту.

Третій розділ деталізує технічний дизайн: вибір архітектурного патерну, підбір технологічного стеку, моделювання реляційної бази даних, проєктування користувацьких інтерфейсів.

Четвертий розділ містить ідентифікацію проєктних ризиків із планами їх мінімізації та поетапний алгоритм пілотного впровадження системи на базі підприємства з подальшою стратегією виходу на комерційний ринок.

Практичне значення результатів підтверджується розробленим архітектурним дизайном, прототипами веб-інтерфейсів та схемою бази даних у форматі Prisma.

Результати роботи були підготовлені до пілотного впровадження на підприємстві ТОВ «ЕЗІК» з подальшим масштабуванням на IT-компанії малого та середнього бізнесу.

Робота містить 110 сторінок, 21 рисунок, 20 таблиць, 1 додаток та 26 використаних інформаційних джерел.

Ключові слова: управління проєктами, управління рішеннями, життєвий цикл рішення, B2B SaaS, Decision Intelligence, штучний інтелект, управління ризиками, інформаційна система.

ТАБЛИЦЯ СКОРОЧЕНЬ ТА ПОЯСНЕНЬ

Скорочення	Пояснення
ШІ	Штучний інтелект
ІС	Інформаційна система
БД	База даних
SME	Small and Medium Enterprises – підприємства малого та середнього бізнесу
B2B	Business-to-Business – бізнес для бізнесу (модель продажу корпоративним клієнтам)
SaaS	Software as a Service – програмне забезпечення як послуга
DI	Decision Intelligence – інтелектуальне управління рішеннями
BI	Business Intelligence – бізнес-аналітика
FSM	Finite State Machine – скінченний автомат
DAG	Directed Acyclic Graph – спрямований ациклічний граф
DLM	Decision Lifecycle Model – розроблена модель життєвого циклу управлінського рішення
ADR	Architecture Decision Records – записи архітектурних рішень
LLM	Large Language Model – велика мовна модель
WBS	Work Breakdown Structure – ієрархічна структура робіт
EVM	Earned Value Management – метод освоєного обсягу
PV	Planned Value – планова вартість запланованих робіт
EV	Earned Value – освоєний обсяг (планова вартість фактично виконаних робіт)
AC	Actual Cost – фактичні витрати на виконання робіт
CPI / SPI	Cost Performance Index / Schedule Performance Index – індекси виконання за вартістю та за графіком
KPI	Key Performance Indicator – ключовий показник ефективності
NPV	Net Present Value – чиста приведена вартість
ROI	Return on Investment – окупність інвестицій
NPS	Net Promoter Score – індекс лояльності клієнтів
MRR	Monthly Recurring Revenue – регулярний місячний дохід

CAPEX	Capital Expenditure – капітальні витрати (інвестиції у створення продукту)
OPEX	Operational Expenditure – операційні витрати (регулярні витрати на підтримку)
API	Application Programming Interface – інтерфейс програмування застосунків
UX/UI	User Experience / User Interface – досвід користувача / інтерфейс користувача
GCP	Google Cloud Platform – хмарна платформа Google
CI/CD	Continuous Integration / Continuous Deployment – безперервна інтеграція та розгортання
MVP	Minimum Viable Product – мінімально життєздатний продукт
ICP	Ideal Customer Profile – ідеальний профіль клієнта
FTE	Full-Time Equivalent – еквівалент повної зайнятості співробітника

ВСТУП

На сьогоднішній день індустрія інформаційних технологій функціонує в умовах безпрецедентної динаміки та швидкості, де від якості управлінських рішень безпосередньо залежить розвиток організації. IT-команди щодня ухвалюють десятки рішень – від стратегічних (вибір технологічного стеку, визначення архітектури, зміна бізнес-моделі) до операційних (пріоритизація спринту, розподіл ресурсів, вибір підходу до реалізації компонента). Кожне з цих рішень має стратегічні наслідки для подальшого розвитку продукту та бізнесу. Проте, незважаючи на критичну роль цих рішень у формуванні продукту та бізнесу, індустрія стикається з фундаментальною проблемою: процеси управління самими рішеннями залишаються неформалізованими, фрагментованими та інструментально не забезпеченими.

Попри розвинену екосистему інструментів для управління задачами, комунікаціями та документацією, у більшості організацій відсутня відповідь на ключове запитання: чому було ухвалено конкретне рішення та до яких наслідків саме це рішення призвело. Це явище має назву «вакуум рішень» (Decision Void), воно призводить до системної втрати інституційної пам’яті підприємства.

За оцінками провідних аналітичних агенцій, глобальний ринок Decision Intelligence демонструє стійку динаміку зростання. За даними Fortune Business Insights, його обсяг становитиме 19,38 млрд доларів США у 2025 році з прогнозом зростання до 57,75 млрд доларів до 2032 року за сукупного річного темпу зростання (CAGR) на рівні 16,9%. Інші аналітичні агенції (Grand View Research, SNS Insider, Precedence Research) наводять близькі прогнози в діапазоні CAGR 15,4–19,3%, що свідчить про галузевий консенсус щодо стрімкого зростання попиту на спеціалізовані інструменти управління рішеннями у корпоративному середовищі.

Наслідки «вакууму рішень» є особливо відчутними для малих та середніх IT-компаній – нові фахівці витрачають тижні на розуміння

історичних причин поточної архітектури, команди повторно обговорюють вже закриті питання через відсутність зафіксованого обґрунтування, виникає критична залежність від засновника чи ключового лідера, який є єдиним носієм контексту (проблема Founder Bottleneck), рішення різних рівнів вступають у суперечність одне з одним через відсутність трасованості залежностей. Аналіз діяльності реальних IT-компаній, проведений під час науково-дослідної практики, підтверджує системний характер цих проблем та обґрунтовує необхідність розробки спеціалізованої інформаційної системи для їх вирішення.

Водночас існуючі теоретичні підходи до формалізації процесу прийняття рішень (Architecture Decision Records, фреймворк RAPID від Bain & Company, модель Cynefin Д. Сноудена) є фрагментованими та не пропонують цілісного інструментального рішення. Жоден з них не розглядає управлінське рішення як керовану і самостійну сутність із власним життєвим циклом, що підлягає моніторингу, переоцінці та аналізу впливу.

Актуальність обраної теми дослідження полягає у необхідності системного аналізу проблематики управління рішеннями в IT-компаніях та розробки нових моделей, методів та інструментів, що забезпечать фіксацію, трасованість та моніторинг управлінських рішень на всіх етапах їх життєвого циклу.

Метою проєкту є планування розробки інформаційної системи для підтримки прийняття управлінських рішень в IT-компаніях Decision Intelligence Platform «DECY» (далі – «DECY») силами команди ТОВ «ЕЗІК» на основі системного аналізу проблематики, математичного моделювання ключових процесів, проєктного планування та планування пілотного впровадження.

Для досягнення поставленої мети визначено такі *завдання дослідження*:

1. Проаналізувати поточний стан процесів прийняття та супроводу управлінських рішень у ТОВ «ЕЗІК» та компаніях контрагентів (яким

надаються послуги) під час надання послуг IT-консалтингу та розробки програмного забезпечення, виявити системні проблеми та слабкі місця.

2. Дослідити світовий ринок Decision Intelligence, провести конкурентний аналіз існуючих рішень та обґрунтувати комерційну і технічну доцільність розробки інформаційної системи «DECY»;
3. Розробити математичні моделі ключових процесів системи:
    а) модель життєвого циклу управлінського рішення на основі апарату скінченних автоматів (FSM);
    б) модель графу залежностей рішень з використанням спрямованих ациклічних графів (DAG).
4. Сформувати план управління проєктом створення інформаційної системи Decy, що включає:
    а) ієрархічну структуру робіт (WBS);
    б) календарний план із визначенням критичного шляху;
    в) структуру команди;
    г) формування кошторису;
    д) проєктування системи контролю виконання на основі методу освоєного обсягу (EVM).
5. Спроєктувати концептуальну IT-архітектуру платформи, логічні моделі бази даних та алгоритми взаємодії користувачів із системою.
6. Ідентифікувати ризики проєкту, спланувати методи реагування на них, розробити план пілотного впровадження системи «DECY» в операційні процеси ТОВ «ЕЗІК» та сформувати стратегію подальшого масштабування на зовнішніх клієнтів.

Об'єктом дослідження виступають процеси прийняття, фіксації та супроводу управлінських рішень в IT-компаніях.

Предметом дослідження є моделі, методи та інформаційні технології підтримки прийняття управлінських рішень в IT-компаніях, а також процеси управління проєктом розробки відповідної інформаційної системи.

Методи дослідження: на першому етапі дослідження було застосовано метод вивчення та критичного аналізу наукових публікацій, галузевих звітів та існуючих фреймворків у сфері Decision Intelligence та управління IT-проєктами. Для комплексного аналізу проблематики використано методи системного аналізу (побудова дерева проблем та дерева цілей, логіко-структурна схема), порівняльного аналізу (конкурентне дослідження ринку), стратегічного аналізу (SWOT-аналіз, модель п'яти сил М. Портера). На етапі математичного моделювання застосовано апарат теорії скінченних автоматів для формалізації життєвого циклу рішення та теорію графів (спрямовані ациклічні графи) для моделювання залежностей між рішеннями. Для планування проєкту використано методи декомпозиції WBS, метод критичного шляху та методи оцінки вартості проєкту.

Наукова новизна отриманих результатів полягає в наступному:

- Розроблено комплексну модель життєвого циклу управлінського рішення Decision Lifecycle Model для оптимізації управління проєктами, яка, на відміну від існуючих підходів, таких як ADR, RAPID чи Cynefin, формалізує рішення як керовану сутність із семи послідовними стадіями та механізмами автоматичного моніторингу на основі апарату скінченних автоматів.
- Удосконалено підхід до аналізу взаємозалежностей управлінських рішень для управління проєктами шляхом формалізації їх причинно-наслідкових зв’язків у вигляді спрямованого ациклічного графу DAG, що дозволяє здійснювати кількісну оцінку каскадного впливу змін та виявляти критичні вузли в структурі рішень організації.
- Запропоновано та розглянуто декілька принципово нових методів для якісного покращення процесу прийняття рішень у IT-компаніях при управлінні проєктами, зокрема сценарії автоматизованого аналізу ризиків, оцінки каскадного впливу та валідації рішень на відповідність корпоративним цінностям і стратегії компанії.

Практичне значення отриманих результатів полягає в тому, що розроблені моделі, архітектурні рішення та проєктна документація формують готове підґрунтя для повноцінної розробки та виведення інформаційної системи «DECY» на B2B-ринок. Теоретичні моделі та методичні рекомендації, сформульовані за результатами дослідження, можуть бути застосовані іншими IT-компаніями малого та середнього бізнесу для оптимізації власних процесів управління рішеннями. Конкретні архітектурні рішення, дизайн інтерфейсів та проєктна документація платформи «DECY» є інтелектуальною власністю ТОВ «ЕЗІК».

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ УПРАВЛІНСЬКИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ ПРОЄКТУ

1.1 Аналіз проблеми неформалізованості процесів управління рішеннями в розрізі управління проєктами в сучасних IT-компаніях

Сучасна IT-індустрія працює в умовах безперервного впровадження змін та ітеративного розвитку продуктів, що суттєво змінило характер управлінської діяльності. Якщо у традиційних галузях керівник може протягом тижня ухвалювати декілька значущих рішень, то в IT-компанії, особливо в сегменті малого та середнього бізнесу, щоденний обсяг рішень вимірюється десятками а іноді і сотнями. Ці рішення охоплюють широкий спектр: від високорівневих (вибір технологічного стеку, дизайн архітектури, зміна пріоритетів продукту) до операційних (пріоритизація задач у спринті, розподіл навантаження між інженерами, визначення підходу до реалізації окремого компонента).

Суттєвою особливістю IT-середовища є те, що навіть локальні технічні рішення мають довгий «хвіст» наслідків. Обраний спосіб побудови API, структура бази даних, інтеграція специфічного компонента або сервісу, модель взаємодії мікросервісів, розміщення UI-елементів – кожне з цих рішень після реалізації стає фундаментом, на якому вибудовуються наступні рішення інших команд. Архітектурні рамки, які задані неявно чи неоднозначно, породжують паралельні компенсаційні реалізації у суміжних командах, кожна з яких робить власні припущення про початковий намір [22]. Тимчасові рішення (workarounds), ухвалені як вимушений компроміс, мають тенденцію ставати сталими практиками, утворюючи своєрідний «інституційний баласт». Він починає диктувати умови для всіх наступних дій команди. Головна проблема непрозорого рішення полягає не в технічних витратах на його рефакторинг, а в каскадному нашаруванні нових компромісів та неявних домовленостей, які доводиться створювати поверх нього.

У більшості IT-організацій наявні системи управління задачами (Jira, Linear), платформи комунікації (Slack, Microsoft Teams), wiki-системи (Confluence, Notion), однак жодна з них не має своїм профільним призначенням фіксацію, трасування та супровід самих рішень як самостійної керованої сутності. Рішення губляться у тредах Slack, коментарях до Jira-тикетів, у пам'яті окремих людей. Саме ця невідповідність між високою щільністю рішень та відсутністю спеціалізованого інструментарію для їхнього управління є стартовою точкою проблематики, яку досліджує ця робота.

1.1.1 Борг рішень в управлінні IT-проєктами

Для формалізованого опису наслідків зазначеної невідповідності в галузевій літературі використовується концепт «decision debt». Борг рішень – це накопичення додаткових витрат на координацію, повторні пояснення, верифікацію та невпевненість, що виникають, коли ключові рішення залишаються невідомими, неоднозначними або нестабільними [22]. Це проявляється у поведінці людей: у повторних обговореннях тих самих питань, у накопиченні коригувальних дій різних команд, у загальній обережності, яка стримує продуктивність.

Саме той факт, що борг рішень проявляється через поведінку людей, робить decision debt особливо небезпечним. Він рідко потрапляє до панелей керівників, не відображається в метриках, проте постійно споживає операційну ємність команди. Одне непрозоре рішення може породжувати накладні витрати, кратно збільшуючи загальну вартість проєкту та формуючи в організації хронічну культуру невпевненості.

Емпіричні дані підтверджують масштаб цього явища. За результатами глобального опитування McKinsey (2019), у якому взяли участь 1259 керівників із 91 країни, лише 20% респондентів зазначили, що їхні організації ухвалюють рішення якісно, а 61% повідомили, що більшість часу, витраченого на прийняття рішень, використовується неефективно [10].

1.1.2 Інституційна амнезія та проблема founder bottleneck

Другим фундаментальним проявом неформалізованості процесів управління рішеннями є інституційна амнезія – систематична деградація організаційної пам'яті. Контекстне знання про причини, альтернативи та компроміси ухвалених рішень здебільшого зберігається не в кодифікованих репозиторіях, а в пам'яті окремих співробітників. Коли ключовий інженер, архітектор чи керівник залишає компанію, разом із ним зникає контекст прийняття рішень, у яких він брав участь. Команди, які залишаються, змушені робити «reverse-engineering» логіки власних систем: відновлювати обґрунтування через повторні обговорення, експерименти, а у гіршому випадку через руйнівні рефакторинги, які згодом показують, що таке рішення було обрано з об'єктивних причин, про які ніхто не пам'ятав, бо вони не були зафіксовані.

Кількісні оцінки втрат від інституційної амнезії є значущими. За даними Phillips Consulting (2024), неефективний обмін знаннями та організаційна амнезія коштують компаніям зі списку Fortune 500 близько 31,5 млрд доларів США на рік у вигляді втраченої продуктивності, дубльованих зусиль та повторних помилок [18]. Орієнтовно 42% критичного корпоративного знання ніколи не фіксується формально, воно існує виключно у пам'яті співробітників і може бути безповоротно втрачене при їхньому звільненні [18]. Компанії середнього та великого розміру втрачають у середньому 4,5 млн доларів на рік на непродуктивних зусиллях з пошуку чи відновлення логіки вже наявних систем [4]. Працівники витрачають понад 100 хвилин щоденно, що еквівалентно приблизно 9 годинам на тиждень, на пошук інформації, необхідної для виконання поточних задач [23].

У IT-компаніях малого та середнього бізнесу інституційна амнезія набуває особливо гострої форми через високий рівень залежності від ключових носіїв контексту – засновників. Формується специфічний патерн, який в галузевій літературі отримав назву founder bottleneck: засновник стає єдиним носієм цілісного бачення того, чому продукт і організація побудовані

саме так, а не інакше. Неприємним наслідком цього явища є той факт, що будь-яке значуще рішення вимагає безпосередньої участі засновника. Це обмежує масштабованість організації та створює критичну залежність від однієї особи, що може поставити під загрозу весь бізнес.

1.1.3 Дефіцит контексту в умовах впровадження ШІ в процеси компанії

Масове впровадження генеративного штучного інтелекту та автономних агентів розробки суттєво змінює процес створення програмного забезпечення. У нових реаліях, коли ШІ здатен генерувати тисячі рядків синтаксично правильного коду за лічені секунди, сам процес програмування перетворюється на базовий загальнодоступний ресурс, а швидкість написання коду остаточно втрачає статус ключової конкурентної переваги. Справжня цінність інженерії повністю змістилася у площину системного дизайну, визначення архітектурних рамок, управління складністю, рішеннями та оркестрацією бізнес-логіки. Як наслідок, слабке місце процесів розробки змістилося від технічної реалізації до розуміння та збереження бізнесового контексту [9].

Великі мовні моделі (LLM), на яких базуються сучасні ШІ-інструменти, ефективно генерують синтаксично правильний код на основі загальнодоступних патернів, проте вони не володіють знаннями про специфіку конкретної організації. Без доступу до формалізованої бази раніше прийнятих управлінських та архітектурних рішень, ШІ генерує рішення, що суперечать рішенням чи компромісам, раніше укладеними командою, оскільки ці вони ніколи не були задокументовані і залишалися виключно в інституційній пам’яті співробітників [25].

Таким чином, формалізація прийнятих рішень (наприклад, через методологію Architecture Decision Records) набуває критичного значення не лише для синхронізації людей, але й для управління діями ШІ. Щоб ШІ-інструменти могли безпечно та коректно працювати в корпоративному ІТ-

середовищі, їм необхідна «база істини» (ground truth) – простір контексту, який містить логіку, обмеження та причини ухвалення попередніх рішень відповідальними фахівцями. Саме тому сучасні компанії потребують інформаційних систем нового покоління, здатних безбар'єрно фіксувати контекст людських рішень та надавати його як структуровані вхідні дані для ШІ-агентів. Без формалізації рішень масштабування розробки з суттєвим використанням ШІ-інструментів стає джерелом системних ризиків та втрати контролю над проєктом.

1.2 Маркетингове дослідження B2B-ринку Decision Intelligence та конкурентний аналіз існуючих рішень

Обґрунтування комерційної та технічної доцільності розробки інформаційної системи «DECY» потребує проведення фундаментального дослідження ринку, на який спрямований продукт, та критичної оцінки існуючих прямих та непрямих альтернатив.

1.2.1 Еволюція від Business Intelligence до Decision Intelligence

Протягом останніх двох десятиліть корпоративна аналітика пройшла шлях від описової Business Intelligence (далі – BI) до прескриптивної та автономної Decision Intelligence (далі – DI). Класичні BI-інструменти (Tableau, Power BI, Qlik) історично фокусувалися на консолідації даних із розрізнених систем та побудови статичних дашбордів, що візуалізують ключові показники ефективності. Проте разом з експоненціальним зростанням обсягу, швидкості та складності корпоративних даних, основним обмеженням стратегічного виконання стала не доступність даних, а когнітивна спроможність людини перетворювати їх на дії.

DI формується як спеціалізована дисципліна на перетині data science, методологій управління проєктами та штучного інтелекту, що має на меті формалізацію, моделювання та часткову автоматизацію життєвого циклу

прийняття рішень. Лише 27% керівників вважають, що їхні організації здатні перетворювати дані на вимірювані бізнес-результати, а 41% цінних спостережень, отриманих із традиційних дашбордів, базуються на даних двомісячної і більшої давності [17]. DI-платформи закривають цей розрив, надаючи валідацію рішень у режимі реального часу та трансформуючи корпоративну архітектуру зі статичної документації на активну операційну інтелектуальну систему.

1.2.2 Конкурентний аналіз існуючих категорій рішень

Попри зростання ринку та чіткий попит, жодна з наявних категорій програмних продуктів не закриває повний спектр потреб управління рішеннями в IT-компаніях. Ринок є фрагментованим за трьома основними категоріями, кожна з яких має суттєві структурні обмеження.

Категорія 1 – Системи управління задачами та колаборативні wiki (Jira, Linear, Confluence, Notion, тощо). Ця категорія є найбільш розповсюдженою серед інструментів у сучасних IT-компаніях. Системи управління задачами оптимізовані для планування та відстеження виконання задач, ведення базової документації та ефемерної комунікації, проте не мають профільного призначення фіксувати обґрунтування ухвалених рішень, яке є ключом для відтворення контексту. Як наслідок, обговорення обґрунтування рішення розчиняються у коментарях до задач (часто зовсім не структуровано), або зовісм не фіксуються. Відокремити рішення від задачі на виконання цього рішення на рівні структури даних в таких системах або неможливо, або непомірно дорого з точки зору затраченого часу.

Колаборативні wiki-платформи (Confluence, Notion) на перший погляд виглядають більш гнучкими: технічно на них можна створити шаблон ADR, завести реєстр рішень у вигляді бази сторінок чи таблиці, встановити посилання між записами та теоретично вибудувати процес, що охоплює весь життєвий цикл рішення – від фіксації та зв’язування із залежними рішеннями до періодичного перегляду та аналізу каскадного впливу змін. Проте wiki є

універсальним інструментом загального призначення, який не передмачає профільної моделі сутності. Це означає, що всю операційну модель управління рішеннями команда змушена вибудовувати та підтримувати власноруч: самостійно проєктувати та підтримувати структуру шаблону, узгоджувати конвенції зв'язування, вручну відстежувати залежності між рішеннями, періодично проводити аудит на предмет застарілих записів, вручну ідентифікувати рішення, які потребують повторного розгляду у зв'язку зі зміною контексту. Wiki не знає, що її сторінка є рішенням; не знає, яке інше рішення залежить від неї; не нагадує про перегляд; не перешкоджає ухваленню нового рішення, що суперечить раніше зафіксованому.

Ключовим обмеженням тут є не відсутність технічної можливості, а надмірна когнітивна та організаційна вартість власноручної розбудови процесів і їх підтримки та адміністрування. В умовах реальних SME-команд, що працюють у швидкому ритмі, додаткова дисципліна, яка не вбудована в інструмент, а вимагає індивідуальних зусиль кожного учасника, закономірно деградує: спочатку пропускаються оновлення, потім фіксація нових рішень, врешті практика припиняється взагалі. Таким чином, навіть за наявності технічного інструментарію wiki залишається пасивним сховищем тексту, а не керованою системою управління рішеннями. Емпіричні свідчення масштабу цієї деградації наведено далі у підрозділі щодо ADR.

Категорія 2 – Платформи Enterprise Architecture, представлені такими продуктами як Bizzdesign та LeanIX, пропонують ретельні фреймворки для узгодження IT-систем із бізнес-стратегією. Однак вони характеризуються високим операційним навантаженням, значним порогом входу для користувачів, структурною ригідністю та високими цінами за ліцензії. Рішення у них трактуються як статична документація, а не як динамічна сутність. Ці недоліки роблять такі системи неприйнятними для сучасних представників малого та середнього бізнесу в IT-індустрії.

Категорія 3 – Спеціалізовані платформи Decision Playbooks (представлені флагманським продуктом Cloverpop) є найближчою за

категорією альтернативою до «DECY» і заслуговують окремого ретельного аналізу. Компанія Cloverpop, заснована у 2012 році в Чикаго, станом на 2025 рік залучила понад 17 млн доларів США інвестицій. Платформа успішно структурує соціальні та колаборативні аспекти управлінських рішень через механізми Decision Trees, Decision Flows, Decision Playbooks та централізованого репозиторію Decision Bank, а також пропонує ШІ-асистент D-Sight для формування рекомендацій на основі корпоративних даних. Попри високу оцінку продукту на ринку, його архітектурні та ринкові характеристики формують декілька структурних обмежень, які відкривають нішу для конкурентного позиціонування «DECY».

По-перше, Cloverpop не орієнтований на IT-сегмент. За офіційними матеріалами компанії, її цільовими галузями є товари широкого споживання (CPG), роздрібна торгівля, фармацевтика та промислове виробництво. Референсні клієнти (Sanofi, Johnson & Johnson, Estée Lauder, Blue Diamond Growers) підтверджують цю спеціалізацію. Пропоновані Cloverpop Decision Playbooks сформовані під типові сценарії цих галузей – закупівля ERP-систем, алокація маркетингового бюджету, тощо.

По-друге, платформа позиціонується виключно для enterprise-сегмента і не орієнтується на малий та середній бізнес. Cloverpop декларує, що їхні клієнти – «найуспішніші світові бренди» (the world's most admired brands), а модель ціноутворення побудована за принципом корпоративного впровадження, яке включає супровід з боку окремо виділенної команди. Для сегмента IT-SME з командами 10–100 осіб, який є цільовим для «DECY», такий підхід є неадекватним як за економічною моделлю, так і за операційною складністю впровадження.

По-третє, продуктовий апарат Cloverpop суттєво відрізняється від того, що пропонує «DECY» та не має під собою математичних моделей, які дозволяють прогнозувати поведінку системи рішень у часі, аналізувати її структурні властивості, проводити каскадний аналіз впливу змін, виявляти

критичні вузли у мережі залежностей, та інших задач, які випливають з цього дослідження.

Узагальнення функціональних можливостей розглянутих категорій та платформи «DECY» наведено в табл. 1.1. Порівняння здійснено за сімома ключовими ознаками, що визначають придатність продукту до задач управління рішеннями в IT-компаніях SME-сегмента.

Таблиця 1.1

Порівняння функціональних можливостей існуючих систем і «DECY»

Функціональна можливість	Jira	Notion	LeanIX	Cloverpop	DECY
Рішення як окрема структурна сутність	Ні	Частково	Так	Так	Так
Формалізований життєвий цикл рішення	Ні	Ні	Так	Так	Так
Типізований граф залежностей між рішеннями	Частково	Частково	Так	Ні	Так
Каскадний аналіз впливу змін	Ні	Ні	Так	Ні	Так
Автоматичні сигнали застарівання рішень	Ні	Ні	Частково	Ні	Так
ШІ-агенти для критики та валідації рішень	Ні	Ні	Ні	Частково	Так
Орієнтація на IT-SME (10–100 осіб)	Так	Так	Ні	Ні	Так

Таблиця 1.1 наочно демонструє структуру виявленої ніші. LeanIX підтримує переважну більшість профільних функцій управління рішеннями, однак залишається недоступним для IT-SME через орієнтацію на великий бізнес та інші сектори економіки, а також через операційну складність. Cloverpop є концептуально найближчим конкурентом, проте поступається за функціоналом – типізованим графом залежностей, каскадним аналізом впливу та автоматизованими сигналами перегляду рішень. Інструменти Jira та Notion, незважаючи на широку популярність у сегменті IT-SME, не мають жодної з

профільних функцій управління рішеннями та використовуються командами не за прямим призначенням.

Узагальнення проведеного конкурентного аналізу також наочно представлено на рисунку 1.1 у вигляді карти позиціонування за двома ключовими вимірами – ступенем спеціалізації продукту під задачі управління рішеннями та рівнем адаптації до сегмента IT-SME.

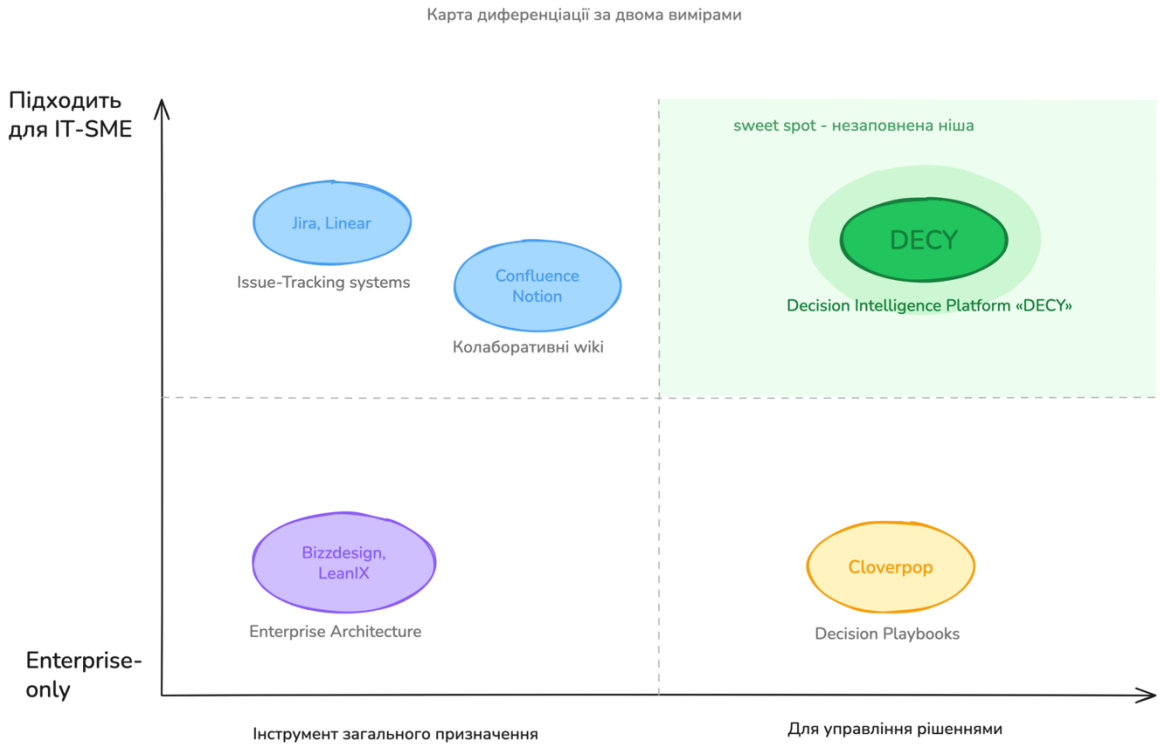

Рис 1.1. Позиціонування категорій на ринку управління рішеннями

Таким чином, «DECY» виходить на ринок не як конкурент у межах однієї з існуючих категорій, а як інформаційна система нового покоління, надаючи інтелектуальну надбудову поверх наявних трекерів задач та вікі-систем, що перетворює рішення з розсіяного контексту на керовану сутність та джерело інституційної пам’яті. Деталізоване обґрунтування концептуального бачення системи та її інноваційності наведено у підрозділі 1.4.

1.3 Декомпозиція проблематики та стратегічний аналіз проєкту

Стратегічний аналіз починається з побудови дерева проблем, яке показує з чого виростає основна проблематика проєкту і до яких наслідків вона призводить для бізнесу. У центрі дерева – неефективність процесів прийняття та супроводу управлінських рішень в IT-компаніях.

1.3.1 Побудова дерева проблем

Розглянемо дерево проблем проєкту (рис. 1.2).

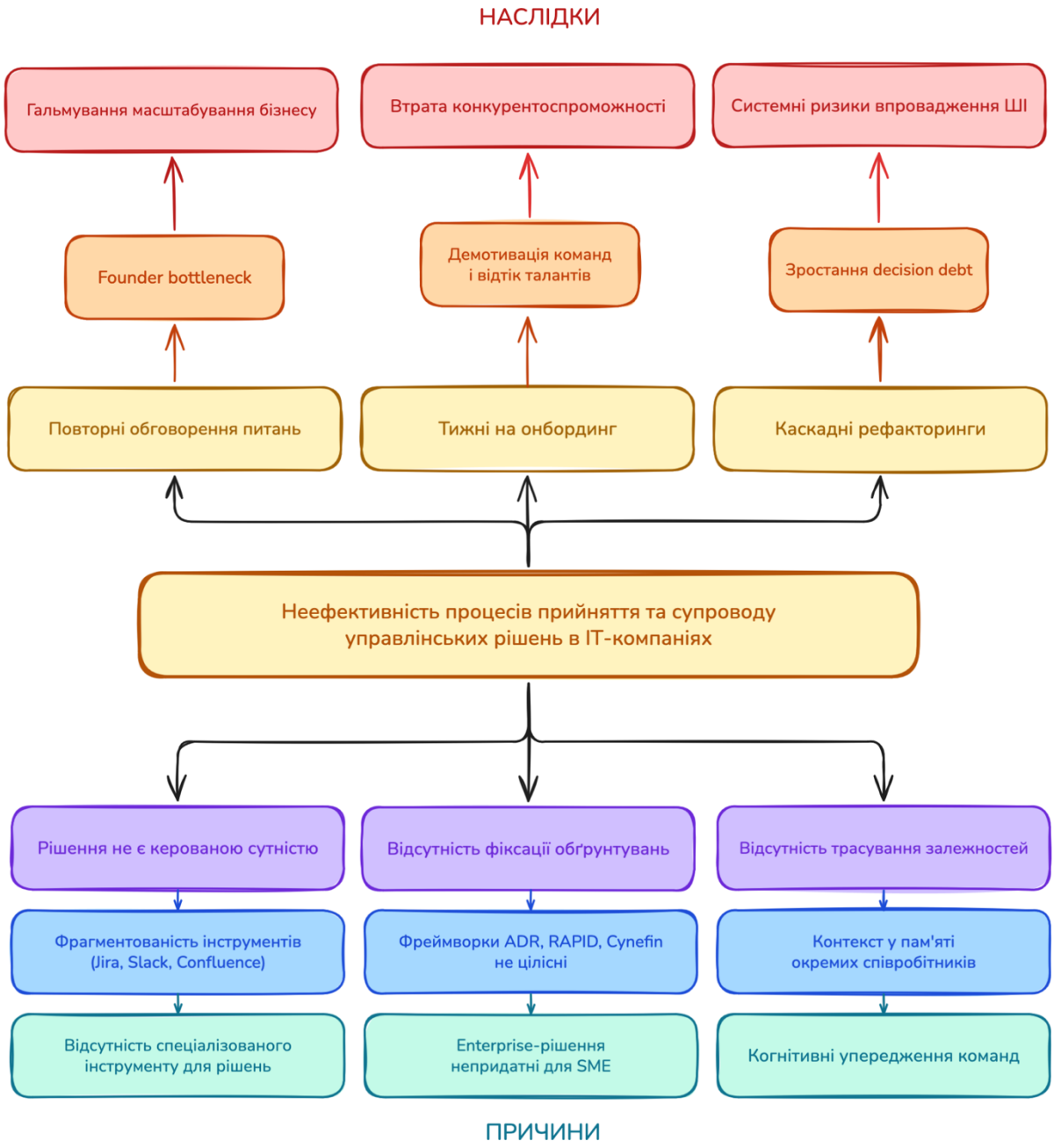

Рис. 1.2. Дерево проблем

Дерево проблем побудовано за методологією логіко-структурного підходу (LFA). Центральну проблему розміщено посередині, її причини нижче у трьох рівнях (від прямих до кореневих), наслідки – вище також у трьох рівнях (від операційних до стратегічних). Стрілки відображають причинно-наслідкові зв’язки.

1.3.2 Побудова дерева цілей та логіко-структурної схеми

Дерево цілей є дзеркальним відображенням дерева проблем: кожне негативне формулювання перетворюється у позитивне. Центральна проблема стає генеральною ціллю проєкту, а причини – засобами її досягнення, наслідки – стратегічними впливами. Причинно-наслідкова ієрархія, виявлена на етапі аналізу проблематики, переходить у ієрархію «засоби ⇒ цілі» для етапу планування.

Логіко-структурна схема (ЛСС) узагальнює результати аналізу у формі матриці, побудованої за чотирма рівнями:

- загальна ціль;
- конкретні цілі;
- результати;
- дії.

Також матриця розбита на чотири виміри для кожного рівня:

- зміст;
- показники досягнення;
- засоби верифікації;
- припущення та ризики.

ЛСС формалізує логіку проєкту та становить основу для подальшого календарного планування, моніторингу виконання та оцінки результатів.

На основі дерева проблем сформовано дерево цілей (рис. 1.3) та логіко-структурну схему (табл. 1.2) проєкту розробки інформаційної системи «DECY».

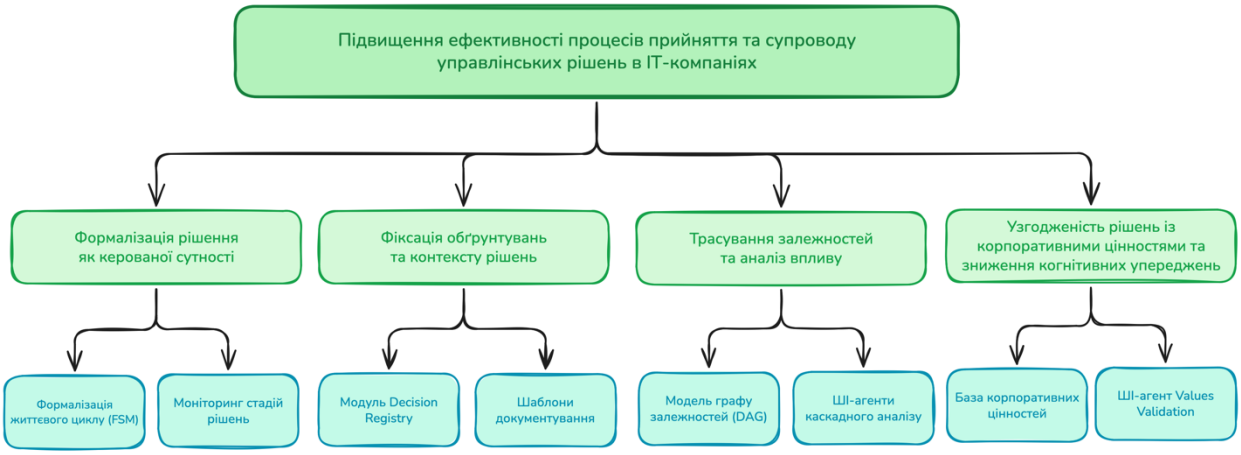

Рис. 1.3. Дерево цілей

Таблиця 1.2

Логіко-структурна схема

Ієрархія цілей	**Показники досягнення**	**Засоби верифікації**	**Припущення та ризики**
1	**2**	**3**	**4**
Загальна ціль: Підвищення ефективності процесів прийняття та супроводу управлінських рішень в IT-компаніях шляхом розробки інформаційної системи DECY	Скорочення часу онбордингу нових фахівців на 40–50%; зниження частки переробок через суперечливі рішення на 30%; 80%+ ухвалених рішень мають зафіксоване обґрунтування і контекст	Метрики пілотного впровадження в ТОВ «ЕЗІК»; опитування команд «до/після»; аналітика ІС DECY	Зберігається попит на інструменти Decision Intelligence на B2B-ринку; ринок IT-SME залишається платоспроможним
Конкретні цілі: 1. Формалізація рішення як керованої сутності з власним життєвим циклом 2.Централізована фіксація обґрунтувань та контексту рішень 3. Трасування залежностей та аналіз каскадного впливу 4. Узгодженість рішень із корпоративними цінностями та зниження когнітивних упереджень	1. 90%+ рішень проходить через формалізовані стадії FSM 2. 80%+ рішень мають заповнений блок Context/Options/Rationale 3. Для 70%+ рішень зафіксовано залежності; 100% критичних рішень проходять cascade analysis 4. 60%+ рішень перевіряються ШІ-агентом Values Validation	Внутрішні метрики ІС DECY; логи ШІ-агентів; аудиторські вибірки рішень; протоколи пілотних спринтів	Команда дотримується процесу фіксації рішень у DECY; корпоративні цінності формалізовані й актуалізуються; якість ШІ-агентів є достатньою для продакшн-валідації

Продовження таблиці 1.2

1	2	3	4
Результати: 1. Математичні моделі FSM (життєвий цикл) та DAG (граф залежностей) 2. MVP платформи DECY: Decision Registry, Graph View, ШІ-агенти 3. База корпоративних цінностей із механізмом валідації 4. Методичні рекомендації для масштабування в SME-сегменті	1. Моделі формалізовано, верифіковано на реальних кейсах 2. MVP покриває 100% функцій першої черги, проходить приймальні тести 3. База містить формалізовані принципи ТОВ «ЕЗІК», інтегрована з агентом валідації 4. Пілот покриває ≥ 50 рішень за 3 місяці; NPS користувачів ≥ 7 5. Документація повна, готова до передачі командам-контрагентам	Розділи 2 та 3 магістерської роботи; технічна документація та репозиторії коду; протоколи приймальних випробувань; план пілотного впровадження	Ресурси команди доступні протягом усього циклу; архітектурні рішення витримують навантаження пілоту; ТОВ «ЕЗІК» готове інвестувати час співробітників; стабільність зовнішніх API (LangChain, LangGraph, LLM-провайдери)
Дії: 1. Дослідження проблематики, ринку, конкурентного ландшафту 2. Розробка математичних моделей FSM і DAG 3. Проєктування архітектури платформи та БД 4. Формування плану проєкту (WBS, календарний план, ресурси, кошторис) 5. Розробка MVP платформи DECY 6. Планування пілотного впровадження в ТОВ «ЕЗІК» 7. Оцінка ризиків та планування стратегій реагування	**Засоби:** Проєктний менеджер (1), бізнес-аналітик (1), архітектор/Tech Lead (1), розробники (2), ML-інженер (1), UX/UI-дизайнер (1), QA-інженер (1); cloud-інфраструктура; ліцензії на LLM-API (OpenAI / Anthropic); робоче обладнання	**Витрати:** Фонд оплати праці команди; оренда cloud-інфраструктури; ліцензії на зовнішні ШІ-API; обладнання; витрати на маркетингову кампанію виходу на ринок	**Передумови:** Доступність команди ТОВ «ЕЗІК»; доступ до емпіричних даних кейсів контрагентів; стабільність SDK LangChain/LangGraph; готовність пілотних учасників надати зворотний зв'язок

Примітка: кількісні значення показників досягнення (зокрема, 40–50% скорочення часу онбордингу, та інші) є цільовими орієнтирами проєкту; методика їх вимірювання, базовий період порівняння (baseline) та критерії досягнення розкриваються у підрозділі 4.2, пунктах 4.2.1 та 4.2.3.

1.3.3 SWOT-аналіз проєкту

SWOT-аналіз є класичним інструментом стратегічного планування, що систематизує фактори реалізації проєкту у двох вимірах: за походженням (внутрішні – сильні та слабкі сторони; зовнішні – можливості та загрози) та за характером впливу (позитивні та негативні). Застосування цього інструменту до проєкту розробки ІС «DECY» дозволяє визначити ключові напрями виведення продукту на B2B-ринок та сформувати основу для планування управління ризиками. Результати аналізу наведено у таблицях 1.3 та 1.4.

Таблиця 1.3

SWOT-аналіз – S & W

Сильні сторони (S) **внутрішні +**	**Слабкі сторони (W)** **внутрішні −**
S1. Наукова новизна. Математична формалізація процесів управління рішеннями на основі FSM та DAG, відсутня у прямих конкурентів	W1. Обмежені фінансові та людські ресурси порівняно з глобальними гравцями ринку DI
S2. Чітка спеціалізація на сегменті IT-SME (10–100 співробітників), незайнята ніша на ринку	W2. Відсутність сталого бренду на міжнародному ринку та публічних референсних клієнтів
S3. Глибока доменна експертиза команди ТОВ «ЕЗІК» у IT-консалтингу та розробці	W3. Ранній етап життєвого циклу продукту, product-market fit ще не перевірено на широкій вибірці
S4. Dogfooding-валідація. Пілотне впровадження в операційні процеси ТОВ «ЕЗІК» як перший референс	W4. Залежність від зовнішніх LLM-провайдерів (OpenAI, Anthropic). Операційні витрати зростають пропорційно до навантаження
S5. Інтеграційна стратегія. «DECY» позиціонується як надбудова над інструментами, які компанії вже використовують – Jira/Linear/Confluence, а не як їх заміна. Низький бар'єр впровадження.	W5. Ризик накопичення технічного боргу під час швидкого розширення функціоналу на post-MVP етапах.
S6. Гнучка цінова модель, адаптована під бюджет IT-SME (на відміну від enterprise-контрактів Cloverpop, Bizzdesign, LeanIX)	W6. Відсутність сформованих sales/marketing процесів та досвіду побудови глобальної воронки B2B-продажів. Обмежений маркетинговий бюджет.

Таблиця 1.4

SWOT-аналіз – O & T

Можливості (O) **зовнішні +**	**Загрози (T)** **зовнішні −**
O1. Стрімке зростання світового ринку Decision Intelligence	T1. Вхід великих гравців (Atlassian, Linear, Notion) з власними decision-модулями на існуючій клієнтській базі
O2. Відсутність прямого конкурента у ніші «DI для IT-SME». First-mover advantage	T2. Можливість виходу Cloverpop у mid-market сегмент або поява нових спеціалізованих стартапів
O3. Масове впровадження ШІ у корпоративні процеси створює попит на схожі системи, здатні надавати ШІ-агентам перевірений бізнес-контекст	T3. Макроекономічна нестабільність (війна в Україні, глобальна рецесія). Скорочення B2B-бюджетів на нові інструменти
O4. Глобалізація та розподілення команд, які працюють віддалено, посилюють проблематику вакууму рішень та підвищують попит на спеціалізовані рішення	T4. Швидка еволюція ШІ-стеку. Ризик часткового застарівання архітектури на LangChain/LangGraph протягом 2–3 років
O5. Регуляторні вимоги (EU AI Act та подібні) стимулюють попит на audit trail управлінських рішень	T5. Залежність від цінової політики LLM-провайдерів та ризик зміни умов доступу до їхніх API
O6. Доступ до грантових та акселераційних програм для українських IT-стартапів (Horizon Europe, USAID, EEN)	T6. Низький рівень «decision maturity» у частини цільових SME. Проблема вакууму рішень не усвідомлюється явно, що ускладнює цикл продажу
O7. Резиденство Дія.Сіті. Оптимізація податкового навантаження. Правові механізми залучення інвестицій.	T7. Регуляторні вимоги до обробки даних створюють додаткові витрати на compliance
O8. Екосистема MCP та відкритих API-стандартів інтеграції з основними IT-інструментами	T8. Потенційні репутаційні бар'єри для української юрисдикції в окремих ринках

На основі проведеного аналізу сформовано матрицю перехресного впливу SWOT-факторів, що дозволяє визначити конкретні напрями дій у чотирьох полях: SO (сильні сторони × можливості), ST (сильні сторони × загрози), WO (слабкі сторони × можливості) та WT (слабкі сторони × загрози).

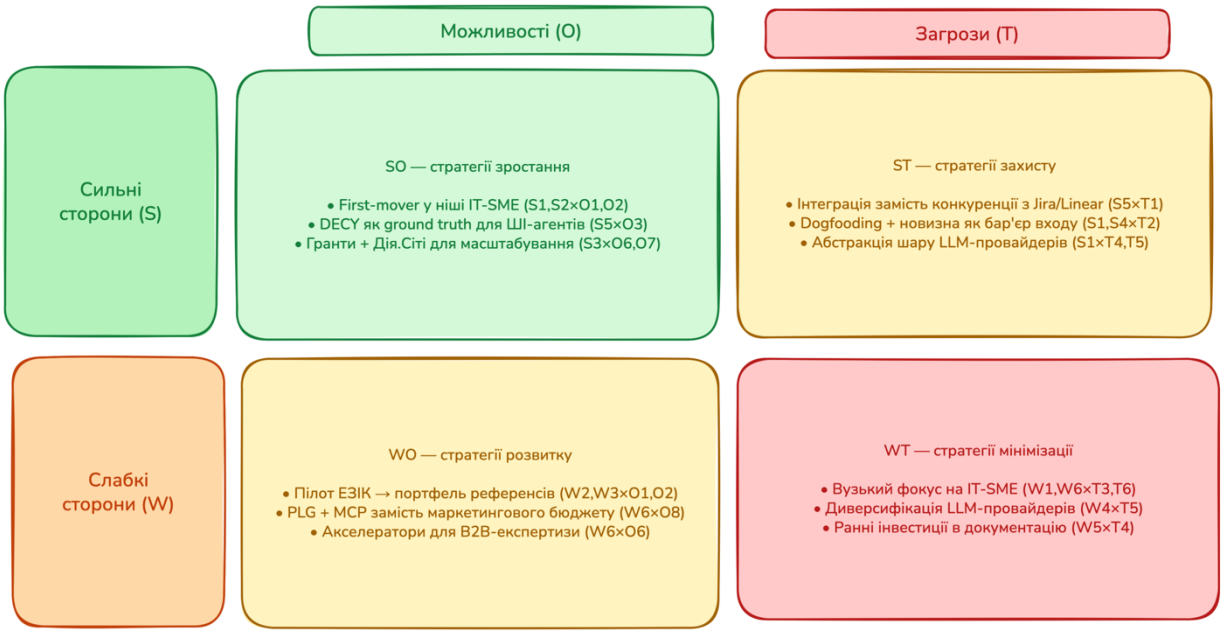

Рис. 1.4. Матриця перехресного впливу SWOT

Поле SO – стратегії зростання. Виведення «DECY» на ринок як першого продукту з математичною формалізацією управління рішеннями у ніші IT-SME до того, як великі гравці розширять власну функціональність (S1, S2 × O1, O2). Позиціонування «DECY» як ground truth системи для корпоративних ШІ-агентів, що використовує масовий попит на структурований бізнес-контекст (S5 × O3). Залучення грантового фінансування (Horizon Europe, USAID) та переваг резидентства Дія.Сіті для прискореного масштабування (S3 × O6, O7).

Поле ST – стратегії захисту. Інтеграційна позиція «DECY» як надбудови над Jira/Linear/Confluence мінімізує пряму конкуренцію з платформами, які можуть розширити власний функціонал decision-менеджменту (S5 × T1). Dogfooding у ТОВ «ЕЗІК» та наукова новизна формують бар’єри входу для нових спеціалізованих стартапів (S1, S4 × T2). Абстракція інтеграції з LLM-провайдерами у окремий архітектурний прошарок знижує залежність від конкретного постачальника (S1 × T4, T5).

Поле WO – стратегії розвитку. Використання пілоту в ТОВ «ЕЗІК» як першого референсу з подальшою побудовою портфелю публічних кейсів для компенсації відсутності впізнаваності бренду (W2, W3 × O1, O2). Модель

Product-Led Growth, а також інтеграція з екосистемою MCP та відкритими API-стандартами забезпечує органічну присутність «DECY» в каталогах інструментів, що частково компенсує обмежений маркетинговий бюджет (W6 × O8). Залучення акселераційних програм для доступу до міжнародної експертизи у побудові B2B-воронки (W6 × O6).

Поле WT – стратегії мінімізації. Фокус на вузькому сегменті IT-SME з чіткими сценаріями впровадження для компенсації обмежених ресурсів та зниження ризиків невдалого сприйняття ринком (W1, W6 × T3, T6). Диверсифікація LLM-провайдерів та моніторинг альтернативних моделей для зниження одночасного впливу W4 і T5. Інвестиції у технічну документацію на ранніх етапах для запобігання накопиченню технічного боргу (W5 × T4).

Отримані результати формують основу для подальшого аналізу ринкової структури за моделлю п'яти сил М. Портера та визначення конкурентної стратегії проєкту.

1.3.4 Аналіз галузі за моделлю п’яти сил М. Портера

Для оцінки конкурентного середовища та визначення стратегії виходу платформи «DECY» на ринок Decision Intelligence використано модель п'яти конкурентних сил М. Портера. Зважаючи на специфіку індустрії програмного забезпечення (SaaS), класичну якісну модель було адаптовано шляхом введення кількісної оцінки факторів впливу (за шкалою від 1 до 5, де 1 – мінімальний вплив/ризик, а 5 – критичний), що дозволяє вивести загальний індекс інтенсивності конкуренції. Оцінка кожної сили є медіанним значенням оцінок її ключових факторів впливу.

1. Загроза появи нових гравців – Оцінка: 3/5

Класичні бар’єри входу (потреба у великому початковому капіталі, доступ до каналів дистрибуції, тощо) у сучасній IT-індустрії суттєво знижені завдяки хмарним технологіям.

– Інженерні та інфраструктурні затрати – доступність відкритих фреймворків (LangChain, LangGraph) знижує вартість розробки базового

функціоналу. Проте створення стійкої математичної моделі на базі скінченних автоматів та графів вимагає глибокої наукоємної експертизи (IP-фактор), що створює технологічний бар'єр. Оцінка загрози: 2/5;

- Маркетингові затрати – ринок B2B-SaaS вимагає значних інвестицій у залучення клієнтів (CAC – Customer Acquisition Cost). Висока вартість маркетингу та циклу прямих продажів є природним бар'єром для нових невеликих команд. Оцінка загрози: 4/5.

2. Загроза появи продуктів-замінників – Оцінка: 4/5

Ця сила є найбільш критичною для «DECY», оскільки цільова аудиторія вже вирішує проблему «вакууму рішень», використовуючи інструменти загального призначення.

- Товари-субститути – Wiki-системи (Confluence, Notion) та системи управління задачами (Jira, Linear);
- Замінники вже інтегровані в процеси IT-компаній, пропонуючи нульову додаткову вартість ліцензій. Однак вони не мають спеціалізованої архітектури для трекінгу рішень, що призводить до прихованих витрат на борг рішень;
- Switching costs (вартість переходу) – перехід від використання загальних wiki до спеціалізованої платформи вимагає не стільки фінансових витрат, скільки зміни корпоративних звичок та впровадження процесів, тобто витрати монжна класифікувати як когнітивні, що робить загрозу замінників високою.

3. Ринкова влада покупців – Оцінка: 3/5

У B2B-сегменті IT-SME покупці є високоінформованими та раціональними.

- Цінова еластичність висока. IT-компанії малого та середнього бізнесу чутливі до операційних витрат і схильні оптимізувати підписки на SaaS-

інструменти, особливо в умовах макроекономічної нестабільності. Оцінка влади: 4/5;

- Довжина циклу прийняття рішення середня. Впровадження інструментів рівня всієї команди вимагає консенсусу між C-level та інженерами, що розтягує цикл продажу, однак невеликі компанії-стартапи зазвичай мають спрощену організаційну структуру та схильні до експериментів. Оцінка влади: 3/5;
- Мережевий ефект утримання максимально високий. Якщо платформа «DECY» стає «базою істини» для компанії, відмова від неї означатиме втрату інституційної пам'яті. Це суттєво знижує владу покупця на етапі після успішного онбордингу. Оцінка влади: 1/5;
- Слід також зазначити, що ринкова влада покупців асиметрична в часі: помірно висока на етапі продажу та критично знижується після успішного онбордингу завдяки мережевому ефекту утримання.

4. Ринкова влада постачальників – Оцінка: 3/5

В контексті розробки ШІ-рішень ключовими постачальниками виступають провайдери хмарної інфраструктури (AWS, GCP) та розробники базових великих мовних моделей, такі як OpenAI, Anthropic, Google.

- Ринок foundation-моделей є олігополістичним. Постачальники мають абсолютну владу над ціноутворенням своїх API;
- Ризик Vendor Lock-in. Висока залежність бізнес-моделі «DECY» від вартості токенів LLM. Стратегія пом'якшення цього впливу передбачає розробку абстрактного архітектурного шару для швидкого перемикання між різними LLM-провайдерами без зміни ядра системи;
- Однак основні модулі системи можуть функціонувати і без ШІ, що суттєво знижує загальний ризик.

5. Інтенсивність конкурентної боротьби – Оцінка: 2/5

– Рівень прямої конкуренції низький. Ринок Decision Intelligence для IT-SME наразі перебуває на стадії раннього формування (Early Adopters);
– Прямі конкуренти (платформи на кшталт Cloverpop або Bizzdesign) сфокусовані виключно на Enterprise-сегменті. «DECY» не конкурує за корпорації, пропонуючи легкий, інтегрований та доступний продукт для ніші, яку ігнорують великі гравці.

Розрахунок загального індексу інтенсивності як середнього арифметичного п’яти сил:

$$Total\ Risk\ Score = (3 + 4 + 3 + 3 + 2) / 5 = 3{,}0$$

Показник 3 з 5 можливих свідчить про помірний рівень ризику, де головною загрозою виступають не прямі конкуренти, а звички користувачів (товари-замінники). Вплив постачальників LLM також є помітним, проте пом’якшується архітектурною абстракцією та здатністю ключових модулів системи функціонувати без ШІ.

Згідно з матрицею базових конкурентних стратегій М. Портера, враховуючи цільову орієнтацію на специфічний сегмент ІТ-компаній (10-100 співробітників) та високу наукоємність продукту, для інформаційної системи «DECY» оптимальною є стратегія диференціації у вузькій ніші.

Платформа не прагне досягти лідерства за витратами (Cost Leadership) на масовому ринку, а здобуває конкурентну перевагу шляхом створення унікальної цінності: перетворення розсіяного контексту прийняття рішень на структуровану і зрозумілу систему для людських команд та автономних ШІ-агентів.

Підсумок аналізу наведено на рис. 1.5.

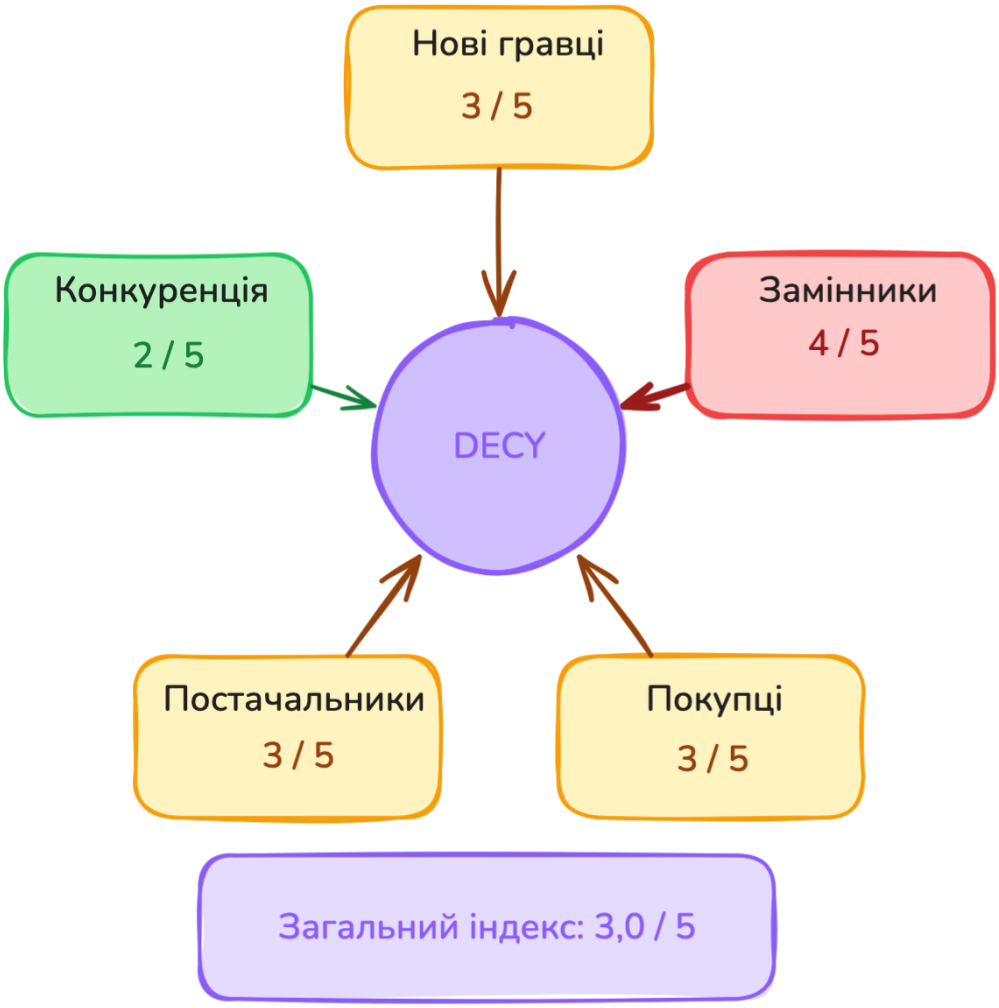

Рис. 1.5. Підсумок аналізу 5 сил Портера

1.4 Формування концептуального бачення інформаційної системи «DECY» та обґрунтування інноваційності IT-проєкту

Результати аналізу проблематики, дослідження ринку Decision Intelligence та конкурентного ландшафту, а також проведений стратегічний аналіз формують базу для концептуалізації інформаційної системи «DECY».

1.4.1 Концептуальне бачення системи

Ключова ідея «DECY» полягає у перетворенні управлінського рішення з неформалізованого побічного продукту комунікації на керовану сутність першого класу з власним життєвим циклом, зафіксованим контекстом, трасованими залежностями та механізмами автоматичного моніторингу. Система будується як спеціалізована надбудова над існуючою екосистемою IT-компанії (Jira, Linear, Confluence, Slack), що замикає проміжок між трекерами задач та wiki-документацією, саме там, де у сучасних IT-компаній виникає вакуум рішень.

Концепцію системи визначають шість фундаментальних принципів:

Принцип 1. Рішення як керована сутність. Управлінське рішення у «DECY» є самостійною сутністю з чітко визначеною структурою (контекст, альтернативи, обґрунтування, наслідки), власним ідентифікатором, станом у життєвому циклі та набором зв'язків з іншими рішеннями. Це принципово відрізняє систему від wiki-платформ, де рішення існує як текст у документі, та від трекерів задач, де рішення губляться у коментарях.

Принцип 2. Frictionless capture. Рішення народжуються у різних контекстах: на мітингах, у тредах Slack, під час code review, у обговореннях Jira-тикетів. Будь-яка система, що вимагає окремо «зайти у DECY» та вручну заповнити форму, програє боротьбу за впровадження. «DECY» проєктується з протилежним припущенням: фіксація рішення має бути максимально безболісною. Система підтримує імпорт з ключових джерел, де рішення реально виникають: автоматичне виявлення рішень у транскриптах мітингів, одноклікова конвертація Slack-повідомлення у чернетку рішення, імпорт з коментарів Jira/Linear, browser-extension для веб-інтерфейсів. Мета полягає у тому, щоб зафіксувати рішення було не дорожче, ніж його не фіксувати.

Принцип 3. Конфігурований життєвий цикл. Різні організації мають власну культуру ухвалення рішень: одні потребують формального підтвердження від C-level, інші працюють за принципом «disagree and commit», треті використовують легкі ADR-процеси. Нав'язування єдиного

фреймворку з обов'язковими стадіями призвело б до того, що «DECY» працював би лише для обмеженого кола команд. Натомість система пропонує конфігурований життєвий цикл. За замовчуванням доступна універсальна модель на основі скінченного автомата (див. розділ 2) з базовими стадіями, проте команди можуть адаптувати набір стадій, умови переходів між ними та правила автоматичного моніторингу під власні операційні процеси.

Для зниження порога входу система постачається з бібліотекою готових шаблонів життєвого циклу під типові патерни організацій. Команди можуть обрати найближчий шаблон як стартову точку та адаптувати його ітеративно. Математичний апарат скінченних автоматів природно підтримує таку модель, кожен шаблон та кожна його адаптація є окремим автоматом, що реалізує загальний інтерфейс життєвого циклу.

Принцип 4. Трасованість залежностей. Рішення утворюють спрямований ациклічний граф (DAG), у якому ребра фіксують причинно-наслідкові зв'язки між ними. Це дозволяє здійснювати каскадний аналіз впливу змін. При перегляді одного рішення система автоматично виявляє всі рішення, що залежать від нього, та пропонує відповідні актуалізації.

Принцип 5. Інтеграційне, а не замінне позиціонування. «DECY» не витісняє інструменти, які команди вже використовують, а збагачує їх. Двосторонні інтеграції з Jira/Linear (зв'язок задач із рішеннями), Confluence/Notion (синхронізація документації) та Slack (фіксація рішень безпосередньо з обговорень) мінімізують операційний поріг впровадження та підтримують принцип frictionless capture.

Принцип 6. AI-augmented прийняття рішень. Архітектура системи передбачає інтелектуальних агентів, побудованих на фреймворках LangChain/LangGraph, що доповнюють людський процес прийняття рішень: генерують контраргументи, прогнозують каскадні наслідки та валідують рішення на відповідність корпоративним цінностям. Таким чином «DECY» виступає не лише системою фіксації, а й інструментом підвищення якості рішень на етапі їх ухвалення.

1.4.2 Обґрунтування іноваційності ІТ-проєкту

Наявні підходи (ADR, RAPID, Cynefin) фіксують рішення, розподіляють ролі або класифікують контекст, але жоден з них не розглядає рішення як об’єкт, яким можна керувати у часі. Новизна роботи полягає у трьох аспектах:

1. ***Рішення як об’єкт зі станом.***

Замість статичного запису (як в ADR) рішення формалізується як скінченний автомат із власним життєвим циклом та функцією спадання актуальності, що дозволяє автоматично ідентифікувати застарілі рішення.

2. ***Залежності як першокласна структура даних.***

Замість гіперпосилань у wiki сукупність рішень моделюється як DAG із типізованими ребрами. Це відкриває каскадний аналіз впливу та виявлення критичних вузлів через алгоритми. Ці можливості відсутні у конкуруючих продуктах.

3. ***ШІ-агенти як активні учасники, а не функція пошуку.***

На відміну від AI-функцій у Notion чи Confluence, де основний фокус полягає у отриманні, підсумовуванні та модифікації даних, які вже існують у системі, у роботі запропоновано агентів, що втручаються у процес ухвалення рішення до його фіксації.

РОЗДІЛ 2. МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ПЛАНУВАННЯ ПРОЄКТУ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ «DECY»

2.1 Модель життєвого циклу управлінського рішення на основі скінченних автоматів

Скінченний автомат (Finite State Machine, FSM) – один з фундаментальних інструментів дискретної математики, що описує поведінку системи через скінченну множину станів та правила переходів між ними у відповідь на події [16]. Класичний аппарат сьогодні застосовується у дуже широкому спектрі задач, таких як проєктування цифрових схем, побудова компіляторів, моделювання бізнес-процесів, тощо.

Для моделювання життєвого циклу управлінського рішення в «DECY» обрано саме FSM з кількох міркувань. Перш за все, апарат забезпечує детермінований опис поведінки, тобто для кожного стану та кожної допустимої події існує рівно один однозначно визначений наступний стан. Це принципово відрізняє FSM від менш формальних нотацій на кшталт BPMN, які допускають неоднозначності та потребують додаткових угод між учасниками для реалізації. У контексті управлінських рішень детермінізм є критичним – якщо рішення перебуває у стані «Затверджено» і отримує подію «Перегляд», система має повести себе передбачувано, без приховування контексту в інтерпретаціях розробника чи інженера.

FSM також природно підтримує верифікацію коректності моделі. Стандартні алгоритми теорії автоматів дозволяють формально перевірити досяжність кожного стану з початкового, відсутність «мертвих» станів, з яких неможливо вийти, та повноту функції переходів. Це дає інженерну впевненість у тому, що модель життєвого циклу спроєктована без пропущених сценаріїв.

Також FSM гарно узгоджується з архітектурою інтелектуальних агентів на базі фреймворку LangGraph (див. розділ 3), який сам по собі реалізований як граф станів. Ця спільна формальна основа забезпечує концептуальну

цілісність системи. Керована сутність (рішення), і агент, що з нею працює, описуються в одній математичній мові.

Слід зазначити одне принципове обмеження класичного апарату. FSM не оперує поняттям часу. Переходи між станами відбуваються миттєво у відповідь на детерміновані події, а сам час знаходження у стані не впливає на поведінку автомата. Для моделювання життєвого циклу рішення це обмеження є суттєвим – рішення, що перебуває у стані «Чинне» протягом двох років без перегляду, з управлінської точки зору поводиться зовсім інакше, ніж щойно затверджене рішення. Тому в роботі запропоновано розширення базової FSM-моделі функцією спадання актуальності.

2.1.1 Формальне визначення моделі Decision Lifecycle Model

Модель життєвого циклу управлінського рішення в «DECY» (далі – DLM) формально визначається як скінченний автомат:

$$M_{DLM} = (Q, \Sigma, \delta, q_0, F), \tag{2.1}$$

де:

- Q – скінченна множина станів життєвого циклу рішення: $Q = \{q_{Draft}, q_{Proposed}, q_{Approved}, q_{InEffect}, q_{UnderReview}, q_{Superseded}, q_{Archived}\}$
- Σ – скінченна множина подій (вхідний алфавит автомата): $\Sigma = \{submit, approve, reject, activate, trigger\_review, supersede, archive, restore\}$
- $\delta: Q \times \Sigma \to Q$ – функція переходів, що визначає, у який стан переходить автомат при отриманні події у поточному стані;
- $q_0 = q_{Draft}$ – початковий стан: будь-яке нове рішення створюється у стані «Чернетка»;
- $F = \{q_{Superseded}, q_{Archived}\}$ – множина термінальних станів, з яких неможливі подальші переходи (за винятком події restore, що повертає рішення до активного циклу).

Семантика станів

Кожен стан моделі відповідає семантично визначеній стадії життя рішення в організації:

- ***Draft (Чернетка)*** – рішення створене, але не сформульоване остаточно; знаходиться в індивідуальній або груповій роботі автора.
- ***Proposed (Запропоноване)*** – рішення сформульоване і подане на розгляд; контекст, альтернативи та обґрунтування зафіксовані.
- ***Approved (Затверджене)*** – рішення пройшло процедуру погодження (відповідно до конфігурації організації) та готове до виконання.
- ***InEffect (Чинне)*** – рішення активно виконується; його наслідки впливають на операції організації.
- ***UnderReview (На перегляді)*** – рішення повторно відкрите для оцінки актуальності; тригером може бути або ручний запит, або автоматичне спрацювання функції спадання актуальності.
- ***Superseded (Замінене)*** – рішення замінене іншим, новішим рішенням; його зміст більше не є дійсним, але історичний запис зберігається для трасованості.
- ***Archived (Архівне)*** – рішення виведено з активного обігу без заміни (наприклад, через втрату актуальності контексту); зберігається лише як історичний документ.

Рішення може переходити з одного стану в інший завдяки функції переходів. Повну специфікацію функції наведено в таблиці 2.1.

Таблиця 2.1

Функція переходів моделі Decision Lifecycle Model

Поточний стан	Подія	Наступний стан	Семантика
1	**2**	**3**	**4**
Draft	submit	Proposed	Автор подає рішення на розгляд
Proposed	approve	Approved	Уповноважена особа/група затверджує рішення
Proposed	reject	Draft	Рішення повертається на доопрацювання
Approved	activate	InEffect	Рішення вступає в силу

Продовження таблиці 2.1

1	2	3	4
InEffect	trigger_review	UnderReview	Автоматичний або ручний запит перегляду
UnderReview	approve	InEffect	Перегляд підтвердив актуальність
UnderReview	supersede	Superseded	Перегляд призвів до заміни новим рішенням
UnderReview	archive	Archived	Перегляд призвів до архівації
InEffect	supersede	Superseded	Пряма заміна без проміжного перегляду
InEffect	archive	Archived	Пряма архівація без проміжного перегляду
Archived	restore	UnderReview	Повернення архівного рішення для повторного аналізу
Superseded	restore	UnderReview	Повернення заміненого рішення для повторного аналізу

Приклад наскрізного проходження рішення

Розглянемо рішення D-017 «Запровадження єдиного Definition of Done для команди розробки» у проєктній команді IT-компанії середнього розміру. Це типове управлінське рішення рівня керівника проєкту, яке стосується критеріїв завершеності задач, потребує погодження з власником продукту і періодично переглядається за результатами ретроспектив та досягнення KPI. Послідовність переходів цього рішення через стани FSM-моделі наведено в табл. 2.2.

Таблиця 2.2

Приклад рішення з типовим проходженням станів

№	Подія	Стан до / після	Actor	Вхідні / вихідні параметри переходу
1	**2**	**3**	**4**	**5**
0	(створення)	0 → Draft	PM	Вхід: ініціатива після ретроспективи. Вихід: id, title, попередній контекст - постійні непорозуміння між розробкою та QA щодо готовності задач
1	submit	Draft → Proposed	PM	Вхід: чернетка рішення. Вихід: alternatives = {єдиний DoD, окремі DoD на тип задачі, неформальні домовленості}; rationale - потреба у спільному розумінні завершеності; уникнення конфліктів трактування

Продовження таблиці 2.2

1	2	3	4	5
2	approve	Proposed → Approved	Product Owner	Вхід: сформульоване рішення з переліком критеріїв (unit-тести, code review, документація, деплой на staging). Вихід: approver = PO; approved_at зафіксовано
3	activate	Approved → InEffect	PM	Вхід: затверджене рішення. Вихід: effective_from - початок наступного спринту; DoD додано в простір команди як обов'язковий чек-лист
4	trigger_review	InEffect → UnderReview	PM	Вхід: метрики поставки за 3 спринти. Вихід: review_reason - стабільне недозакриття спринтів через надмірну строгість DoD для дрібних задач
5	supersede	UnderReview → Superseded	PM	Вхід: висновок ретроспективи. Вихід: superseded_by - нове рішення про дворівневий DoD (спрощений для багфіксів, повний для фіч); D-017 закрите

Побудована модель має такі властивості, які підтверджують її коректність як FSM:

1. ***Детермінізм*** **–** для кожної пари (стан, подія) функція δ визначає не більше одного наступного стану;
2. ***Досяжність*** **–** кожен стан $q \in Q$ є досяжним з початкового стану q_0 за скінченне число переходів;
3. ***Відсутність ізольованих станів*** **–** з кожного стану $q \in Q \setminus F$ існує принаймні один вихідний перехід;
4. ***Контрольована термінальність*** **–** стани $q_{Superseded}$ та $q_{Archived}$ є умовно термінальними, тобто з них можливий лише перехід за виконання події $restore$, що повертає рішення до активного циклу.

Графічне представлення Decision Lifecycle Model у вигляді діаграми станів наведено на рис. 2.1.

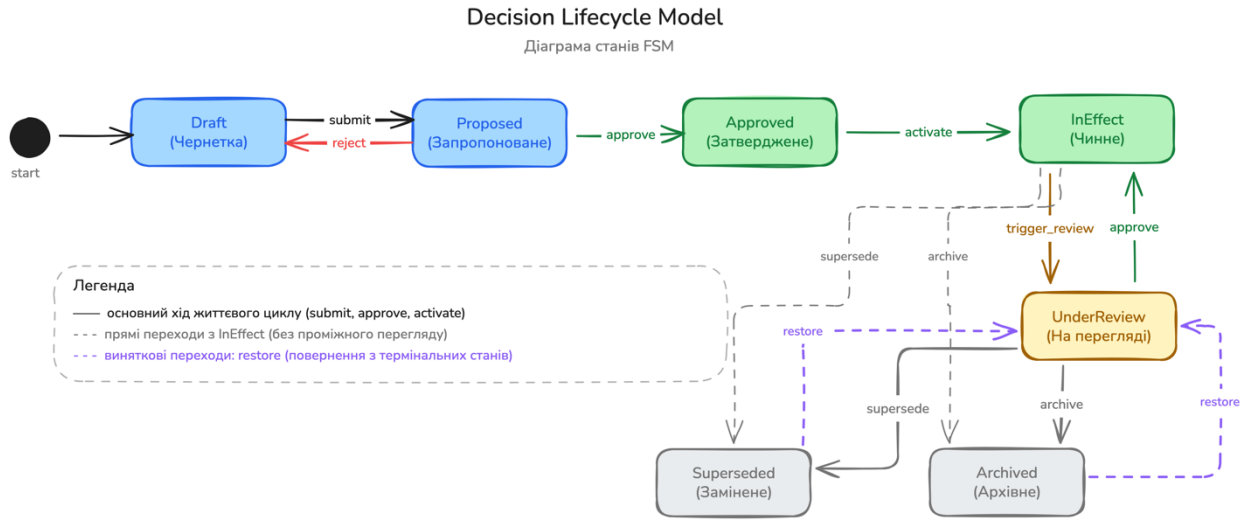

Рис. 2.1. Діаграма станів DLM / FSM

2.1.2 Параметризація моделі під шаблони різних організацій

Базова модель DLM, описана у попередньому підрозділі, формалізує універсальний життєвий цикл управлінського рішення з повним набором стадій. Однак практика показує, що різні організації мають принципово різні культури ухвалення рішень. Наприклад, ранній стартап з командою у дев'ять людей не потребує формальної процедури затвердження, тоді як корпоративна структура з чотирма рівнями менеджменту має чітке розмежування процесу узгодження рішень. Інженерні команди надають перевагу легким ADR-процесам, тоді як кросфункціональні ініціативи частіше застосовують RAPID-подібні моделі з явним розподілом ролей. Нав'язування єдиної моделі всім організаціям призвело б до того, що «DECY» як інформаційна система була б корисною лише для вузького кола команд.

Натомість, пропонується трактувати DLM не як одну фіксовану модель, а як шаблон автомата (template), з якого породжуються конкретні реалізації для конкретних організацій. Шаблон визначає базову структуру, спільну для всіх реалізацій, та простір допустимих модифікацій. Така постановка зберігає математичну строгість моделі (кожна інстанція все ще є коректним

скінченним автоматом) і одночасно дає організаціям необхідну гнучкість конфігурації.

Формально шаблон DLM визначається кортежем:

$$T_{DLM} = \left(Q_{base}, \Sigma_{base}, \delta_{base}, q_0, F_{base}, Q_{opt}, \Sigma_{opt}, R_{mod}\right), \quad (2.2)$$

де:

- $Q_{base}, \Sigma_{base}, \delta_{base}, q_0, F_{base}$ – обов'язкова частина, що присутня в кожній інстанції шаблону. Для DLM обов'язковими станами є $Draft$, $InEffect$, та $Superseded$. Без них рішення не може існувати як керована сутність взагалі;
- Q_{opt} – множина опціональних станів, які інстанція може включити або виключити: $Proposed$, $Approved$, $UnderReview$, $Archived$;
- Σ_{opt} – множина опціональних подій, що відповідають опціональним станам;
- R_{mod} – набір правил модифікації, які гарантують, що результуюча інстанція залишиться коректним FSM.

Правила модифікації R_{mod} обмежують, які саме конфігурації допустимі:

- З початкового стану q_0 повинен існувати шлях до стану $q_{InEffect}$, без цього модель не має сенсу, бо в такому випадку рішення ніколи не вступить в силу;
- При додаванні чи видаленні станів функція переходів δ автоматично перебудовується, але результуюча функція повинна залишатися детермінованою, тобто для кожної пари (стан, подія) визначений рівно один наступний стан;
- Перед збереженням нової конфігурації система формально перевіряє досяжність кожного стану з q_0,відсутність ізольованих станів та повноту функції переходів. Конфігурація, що порушує ці властивості, відхиляється з повідомленням про конкретну проблему.

На основі описаного вище підходу, в «DECY» постачаються три заздалегідь спроєктовані шаблони, що покривають типові патерни невеликих, маленьких та середніх IT-організацій.

Classic. Повний набір з усіх семи станів базової DLM.

Lightweight Startup. Найпростіший з можливих робочих циклів. Рішення створюється, набуває чинності, у якийсь момент замінюється новим або архівується. Без формального підтвердженняя, без окремої стадії перегляду. Автор сам вирішує, коли його чернетка рішення готова, і просто публікує її. Цей шаблон оптимальний для команд до 5 осіб, де вартість координації перевищує цінність формалізації. Інакше кажучи, де кожен зайвий клік у процесі стає причиною того, що команда взагалі перестає фіксувати рішення.

RAPID. Шаблон базується на моделі розподілу ролей Bain & Company, де кожна стадія асоційована з конкретною роллю – Recommend, Agree, Perform, Input, Decide. Кожна подія несе додатковий атрибут ролі, який фіксує не лише факт переходу, але й те, хто саме у структурі повинен ініціювати конкретний перехід. Це робить шаблон корисним для кросфункціональних ініціатив, де відсутність явного розподілу ролей зазвичай і є головною причиною того, що рішення зависають.

Наведені вище шаблони це лише вже готові конфігурації, які поставляються разом з інформаційною системою. Цей список не є вичерпним. Організації можуть створювати власні шаблони на основі базового. Система автоматично перевіряє валідність нової конфігурації та зберігає її як конфігурацію із заданим найменуванням. Це робить «DECY» придатним для еволюції разом з організацією.

2.1.3 Функція спадання актуальності управлінських рішень в IT-проєктах

Класичний апарат скінченних автоматів має суттєве обмеження для застосування в управлінні рішеннями – він не оперує поняттям часу. Для більшості задач, які традиційно моделюються через FSM, це обмеження не є

проблемним. Але для моделювання життєвого циклу управлінського рішення воно стає критичним. Розглянемо реалістичний сценарій. У 2023 році команда зафіксувала рішення про вибір PostgreSQL як основної бази даних, обґрунтувавши його обсягом даних, що очікувався протягом наступних двох років. Рішення пройшло всі стадії та перейшло у стан $InEffect$. Проходить три роки. Обсяг даних виріс в декілька разів, з'явилися нові вимоги до аналітики, з'явилися нові класи бази даних, яких на момент ухвалення не існувало. Базова FSM-модель не дає жодного сигналу. Рішення формально перебуває у стані $InEffect$, з точки зору автомата все нормально. Але з управлінської точки зору рішення вже не актуальне, його обґрунтування здеградувало, передумови, на яких воно ґрунтувалося, більше не діють.

Тому пропонується розширити базову модель функцією спадання актуальності (freshness decay function), що дозволяє автомату знати про час перебування рішення у стані $InEffect$ та автоматично ініціювати перегляд, коли актуальність падає нижче заданого порогу [3].

Актуальність рішення моделюється як функція часу:

$$f(t) = e^{-\lambda(t-t_0)}, \tag{2.3}$$

де t_0 – момент входу у стан $InEffect$, λ – коефіцієнт спадіння. Значення $f(t)$ змінюється від 1 у момент набуття чинності до близьких до нуля значень у далекому майбутньому. Експоненціальна форма обрана як стандартна модель забування з широкою емпіричною підтримкою у суміжних дисциплінах (наприклад у психології пам'яті, системах рекомендації, епідеміології).

Для зручності інтерпретації коефіцієнт λ задається через період напіврозпаду актуальності – час, за який актуальність падає вдвічі. Замість того, щоб думати про абстрактне число λ, користувач оперує зрозумілими термінами: «це рішення стає наполовину менш актуальним приблизно за рік».

Різні класи рішень старіють принципово різними темпами. Архітектурне рішення про вибір мови програмування може залишатися актуальним 5–10 років. Тактичне рішення про запуск конкретного продукту на ринок є актуальним кілька місяців. Операційне рішення про правила роботи команди

актуальні декілька років, а може й менше, в залежності від контексту. Глобальна фіксована швидкість спадання не має сенсу. У «DECY» рішення класифікуються за типами на момент створення, і кожен тип має свою характерну швидкість спадання. Базовий набір типів покриває типові класи рішень в IT-організаціях:

- Управлінські – найм персоналу, управління ресурсами, планування та контроль процесів розробки, формування бюджету та управління закупівлями, вибір стратегій реагування на ризики;
- Архітектурні – вибір технологічного стеку, патернів інтеграції, моделей даних;
- Продуктові – рішення про продуктову стратегію, цільову аудиторію, ключовий функціонал;
- Тактичні – запуск конкретних експериментів, маркетингових кампаній, тимчасових рішень;
- Операційні – правила роботи команди, процеси, ролі та зони відповідальності;
- Інфраструктурні – рішення про конкретні провайдери, конфігурації середовищ, інструментарій.

При створенні організації та первинному налаштуванні, адміністратор обирає типові значення періоду напіврозпаду для кожного конкретного класу рішення. Також, за потреби, можуть бути додані нові типові класи.

Користувач при створенні рішення обирає тип зі списку (опціонально), а система застосовує відповідний коефіцієнт спадання за замовчуванням. Якщо тип не було вказано, система спробує автоматично визначити якій категорії належить рішення застосовуючи ШІ, виходячи з контексту рішення та його даних. Користувач також може змінити значення коефіцієнту спадання актуальності вручну для конкретного рішення за потреби. На рис. 2.2 наведено приклад візуалізації налаштованої функції спадання актуальності. Значення для конфігурації функції було використано лише для демонстрації як може виглядати функція, щоб підкреслити характерні нюанси.

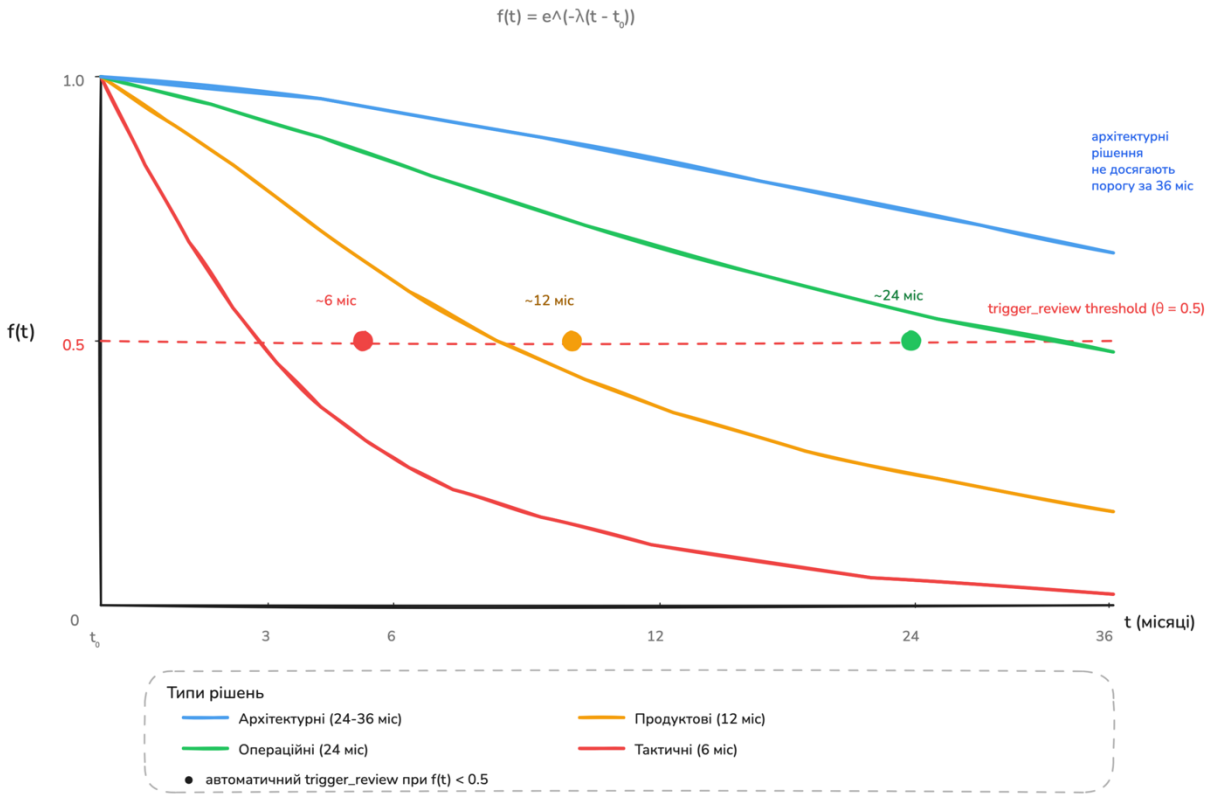

Рис. 2.2. Приклад функції спадання актуальності рішень

2.2 Модель графу залежностей управлінських рішень на основі спрямованих ациклічних графів

Управлінські рішення в реальних організаціях не існують ізольовано. Вони можуть ґрунтуватися на якомусь попередньому рішенні, обмежувати наступні, або перебувати у певному відношенні з паралельними рішеннями. Без формалізації цих зв’язків неможливо ані зрозуміти структуру наявного ландшафту рішень, ані передбачити наслідки змін, ані виявити внутрішні суперечності. Тому друге ключове завдання математичної моделі «DECY» полягає у стовренні граф зв’язків між усіма рішеннями організації.

Для формалізації цієї структури обрано апарат спрямованих ациклічних графів (Directed Acyclic Graphs, DAG). Цей вибір на перший погляд не зовсім очевидний. Може здатися, що звичайного графу або структури дерева буде достатньо, але після детального поглиблення у специфіку цієї проблеми, можна побачити що це зовсім не так.

Список зв'язків. Найпростіше вирішення проблеми зв'язків – зберігати у кожного рішення список ідентифікаторів пов'язаних рішень. Цей підхід використовується у wiki-системах (Confluence, Notion), де зв'язки між сторінками реалізовані як гіперпосилання. Проблема в тому, що такий формат не підтримує запитів структурного характеру: знайти всі рішення, що транзитивно залежать від конкретного; виявити, чи існує суперечність у ланцюгу обґрунтувань; обчислити критичні вузли мережі рішень. Усі ці задачі формулюються на мові теорії графів і потребують саме графової моделі даних, а не просто колекції посилань.

Спрямованість. У зв'язках між рішеннями завжди присутня асиметрія. Рішення А може ґрунтуватися на рішенні В, але не навпаки. Ця асиметрія є хронологічною (В існувало раніше) та логічною (В обґрунтовує А, не А обґрунтовує В). Ненапрямлений граф втратив би цю інформацію, а саме з неї виводяться найважливіші для управління властивості, даючи змогу прогнозувати каскадний вплив, будувати ланцюги аргументації та відстежувати базові передумови будь-якого архітектурного (або іншого) вибору.

Ациклічність. Наявність циклу в графі рішень означала б логічний парадокс: рішення А обґрунтовує В, В обґрунтовує С, а С знову спирається на А. Тобто рішення опосередковано випливає саме з себе. Як і в розробці програмного забезпечення, де циклічні залежності є критичною помилкою архітектури, у менеджменті це свідчить про хибну логіку. Використання спрямованих ациклічних графів (DAG) вирішує цю проблему математично. Система на рівні бази даних просто блокує будь-яку спробу створити зв'язок, що замикає цикл.

Дерево. Дерево є окремим випадком DAG, у якому кожен вузол має не більше одного parent. Для рішень це обмеження надто жорстке. Одне рішення часто ґрунтується одночасно на кількох попередніх. Наприклад, рішення про вибір моделі впровадження може спиратися на архітектурне рішення про мікросервіси, на операційне рішення про розмір команди, та на

інфраструктурне рішення про хмарне середовище. У деревоподібній структурі довелося б штучно обирати основний вузол, втрачаючи інформацію про решту зв'язків. DAG зберігає всю мережу обґрунтувань.

Таким чином, архітектура на базі DAG вирішує ключові проблеми моделювання рішень. Вона значно гнучкіша за звичайні деревоподібні структури, математично унеможливлює виникнення логічних парадоксів і спирається на багату алгоритмічну базу, наприклад пошук у ширину та глибину. Це робить її фундаментальним компонентом для розробки платформи.

2.2.1 Формальне визначення DAG-моделі взаємозв'язків між управлінськими рішеннями в IT-проєктах

Математично запропоновану структуру можна описати дуже просто:

$$G = (V, E), \tag{2.4}$$

де:

- $V = \{v_1, v_2, \dots, v_n\}$ – це скінченна множина вершин графа. Кожна вершина відповідає конкретному рішенню. У контексті системи вершина v_i є не просто ідентифікатором, а комплексним об'єктом, що містить метадані, обґрунтування та поточний статус життєвого циклу;
- $E \subseteq V \times V \times T$ – множина спрямованих ребер, які описують зв'язки між рішеннями. На відміну від класичних графів, кожне ребро у системі «DECY» є типізованим. Воно описується триплетом (u, v, t), тобто рішення u впливає на рішення v через тип зв'язку t.

Типологія зв'язків

Для коректного моделювання бізнес-логіки алфавіт типів зв'язку включає чотири фундаментальні категорії:

1. ***Основа (Basis).*** Найжорсткіший тип залежності. Означає, що рішення u є фундаментальною передумовою для ухвалення рішення v. Наприклад, рішення використовувати PostgreSQL як базу даних обґрунтовує похідне рішення застосування розширення PostgreSQL

«pg_vector» для роботи з векторами. Якщо базове рішення скасовується, похідне рішення автоматично втрачає свою легітимність.

2. ***Залежність (Dependency).*** М'яка форма зв'язку. Рішення v враховує контекст рішення u, але не є критично залежним від нього. Наприклад, рішення щодо стратегії найму інженерів впливає на рішення про сегментацію команд, але зміна стратегії найму не робить структуру команд миттєво неправильною, а лише сигналізує про необхідність її перегляду.
3. ***Заміщення (Supersession).*** Хронологічний зв'язок. Означає, що нове рішення v офіційно замінює застаріле рішення u. Цей тип зв'язку завжди спрямований вперед у часі і створюється автоматично при зміні статусів у FSM-моделі.
4. ***Конфлікт (Conflict).*** Сигнальний зв'язок, який фіксує логічну суперечність між двома рішеннями. Їхнє одночасне виконання є неможливим.

Умова ациклічності та архітектурна чистота моделі

Фундаментальною властивістю ациклічного графа є відсутність замкнених циклів [5]. Проте, з погляду архітектури системи, на ациклічність перевіряються лише ребра типів *Basis*, *Dependency* та *Supersession*, оскільки вони формують причинно-наслідкову логіку. Зв'язки типу *Conflict* є симетричними (якщо А суперечить В, то і В суперечить А) і технічно реалізуються як окрема реалізація поверх основного графа. Це дозволяє зберегти математичну чистоту моделі DAG для алгоритмічного аналізу, одночасно надаючи користувачам гнучкість у фіксації протиріч.

Алгоритмічно система блокує створення будь-якого зв'язку, що призводить до зациклення. Перед додаванням нового ребра від u до v система виконує стандартний пошук у глибину перевіряючи, чи не існує вже зворотного шляху від v до u.

Приклад моделювання фрагмента графа

Для демонстрації можливостей моделі розглянемо спрощений сценарій ухвалених рішень в IT-команді на рис 2.3, де були прийняті наступні рішення:

- v_1 – «Перехід на мікросервісну архітектуру»
- v_2 – «Вибір мови програмування Go для бекенду»
- v_3 – «Використання PostgreSQL як основної БД»
- v_4 – «Впровадження через Kubernetes у Google Cloud»
- v_5 – «Ізольована схема БД для кожного сервісу»
- v_6 – «Перехід на бізнес-модель Pay-per-seat ($29/міс)»

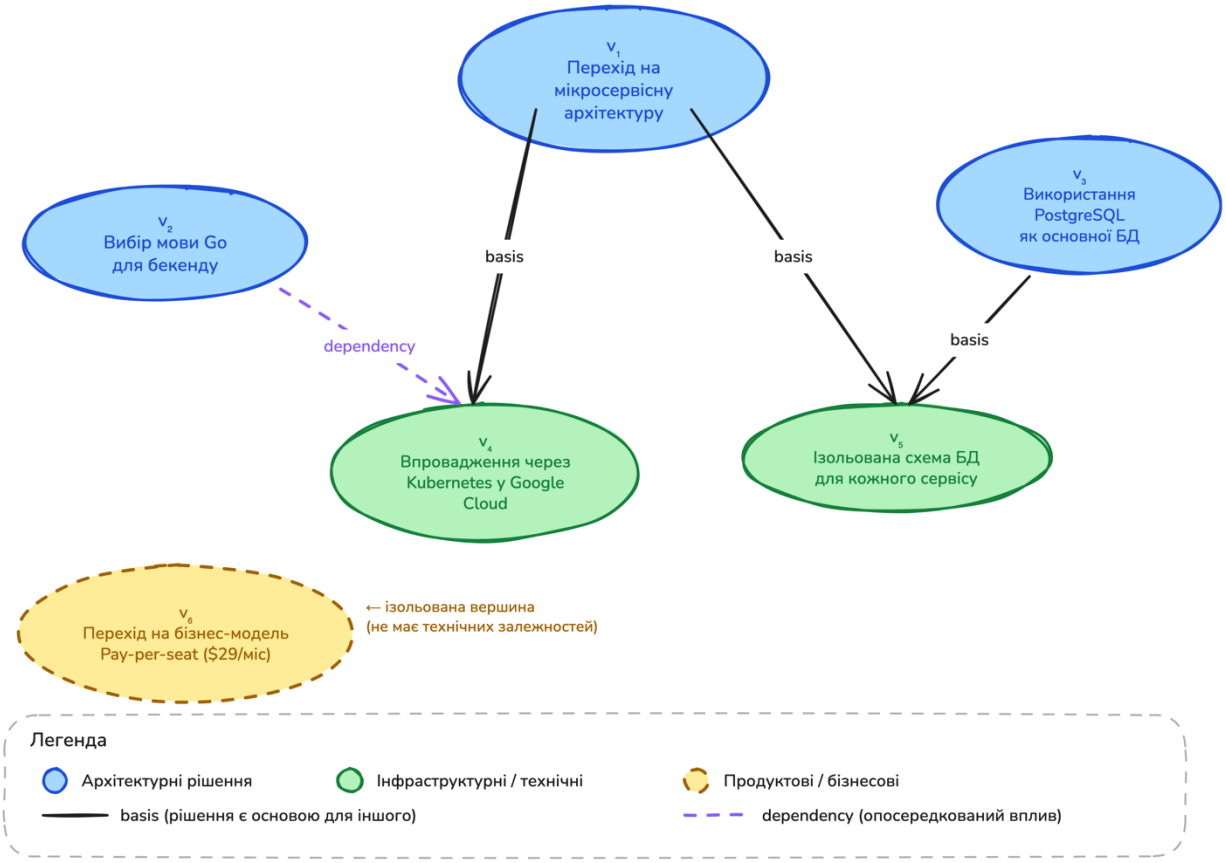

Рис. 2.3. Приклад фрагмента графа залежності рішень

Найбільш показовою точкою у цьому графі є вершина v_5, на неї одночасно вказують два ребра типу basis – від v_1 (мікросервісна архітектура є передумовою для ізоляції схем) та від v_3 (вибір PostgreSQL формує можливості для такої ізоляції). Це і є та сама ситуація множинного

обґрунтування, заради якої був обраний апарат DAG, а не дерева. Одне рішення законно може мати кілька рівноправних основ.

Окремо виділено ребро від v_2 до v_4 з типом *dependency*. Вибір мови програмування Go не є безпосередньою передумовою для впровадження мікросервісів за допомогою Kubernetes (Kubernetes можна обрати незалежно від мови), але має опосередкований вплив на конкретну конфігурацію контейнерів.

Розрізнення цих двох типів ребер (*basis* як логічна основа та *dependency* як побічний вплив) дозволяє на наступних етапах диференціювати каскадний аналіз. Зміна v_1 потребує обов'язкового перегляду v_5, тоді як зміна v_2 лише сигналізує про необхідність перевірки конфігурації v_4, не вимагаючи повного перегляду.

2.2.2 Каскадний аналіз впливу та виявлення критичних рішень

Сформульована DAG-модель безпосередньо підтримує два класи аналітичних запитів, які лежать в основі цінності «DECY» для організації – каскадний аналіз впливу та виявлення критичних рішень.

Каскадний аналіз впливу

При перегляді або зміні рішення u необхідно відповісти на запитання: на які інші рішення це вплине? Формально шукається множина всіх вершин, до яких існує спрямований шлях з u:

$$Impact(u) = \{v \in V : \text{існує шлях } u \to v\} \qquad (2.5)$$

Вплив обчислюється стандартним обходом графа в ширину або глибину зі складністю $O(V + E)$, де V – кількість рішень, E – кількість зв'язків між рішеннями. Навіть якщо в організації накопичиться 10 000 рішень з 50 000 зв'язками між ними, каскадний аналіз для будь-якого рішення обчислюється за лінійний час від розміру графа. Це гарантує, що система залишатиметься швидкою при масштабуванні.

Для ілюстрації розглянемо фрагмент DAG із шести типових управлінських рішень у проєктній команді:

– D1 – методологія управління проєктами (Scrum);
– D2 – стратегія комунікації з замовником (щотижневі демонстрації);
– D3 – тривалість спринту (2 тижні);
– D4 – формат sprint review;
– D5 – структура команди (cross-functional);
– D6 – система командних KPI.

Множину ребер E (відношення *basis*) визначено таким чином:

$$E = \{D_1 \to D_3, D_1 \to D_5, D_2 \to D_4, D_3 \to D_4, D_3 \to D_6, D_5 \to D_6\} \quad (2.6)$$

У даному прикладі обрана методологія є фундаментом для встановлення тривалості спринту та структури команди; стратегія комунікації з замовником разом із тривалістю спринту визначають формат регулярних демонстрацій; тривалість спринту разом зі структурою команди визначають систему командних метрик.

Обчислимо множину $Impact(u)$ для кожного рішення обходом графа в ширину. Результати наведено в табл. 2.3.

Таблиця 2.3

Impact(u) для кожного рішення

Рішення	Прямі наслідки	Усі залежні (транзитивно)	||Impact(u)||
D1	{D3, D5}	{D3, D4, D5, D6}	4
D3	{D4, D6}	{D4, D6}	2
D2	{D4}	{D4}	1
D5	{D6}	{D6}	1
D4	-	-	0
D6	-	-	0

Результати мають конкретну управлінську інтерпретацію. Зміна D1 (методології управління проєктами) автоматично ініціює подію перегляду для чотирьох залежних рішень, тоді як зміна D4 (формату демо) чи D6 (системи KPI) не призводить до жодних каскадних переглядів. Окремо слід зазначити, що зона впливу D1 повністю охоплює зону впливу D3, тобто каскад

поширюється послідовно через ланцюжок від D1 до D3, а далі до D4, і не обмежується лише прямими наслідками. Саме ця транзитивність і відрізняє формальний DAG-аналіз від простого підрахунку прямих залежностей у звичайних wiki-системах. Рішення D1 напряму впливає лише на два інших, але реальний обсяг каскадного перегляду виявляється удвічі більшим.

Виявлення критичних рішень

Наступне аналітичне завдання полягає у визначенні структурної ваги кожного рішення. Перегляд яких саме вузлів призведе до найбільшого каскадного впливу на систему? Для платформи «DECY» найбільш релевантним є алгоритм PageRank з теорії графів. Його рекурсивна природа ідеально відображає специфіку управлінської архітектури. Вага конкретного рішення визначається не лише кількістю його прямих наслідків, а й стратегічною важливістю тих вузлів, які на нього спираються. Іншими словами, рішення стає критичним не тоді, коли генерує багато дрібних залежностей, а коли виступає фундаментом для інших ключових рішень.

З управлінської точки зору, автоматичне ранжування рішень за метрикою PageRank надає керівникам та технічним лідерам корисний інструмент для пріоритезації фокусу уваги. Найбільш важкі вузли графа система автоматично маркує як зони підвищеного ризику, і будь-які пропозиції щодо зміни чи скасування таких рішень вимагатимуть максимального рівня консенсусу, ретельного аудиту та залучення старших фахівців. Рішення з низьким показником центральності, наприклад ізольовані вузли графа можуть безпечно делегуватися автономним командам без зайвого бюрократичного навантаження. Цей підхід вирішує вищезгадану проблему founder bottleneck.

Крім аналізу ризиків, виявлення критичних вузлів графа відкриває нові можливості для управління персоналом, наприклад у процесах адаптації нових розробників. Замість неструктурованого вивчення всієї проєктної документації, інформаційна система «DECY» здатна генерувати оптимальний

маршрут занурення у контекст. Система пропонує новому співробітнику ознайомитися спочатку з тими рішеннями, які мають найвищий рейтинг PageRank. Це гарантує, що інженер насамперед засвоїть фундаментальні архітектурні обмеження та компроміси, на яких тримається вся система, суттєво знижуючи ризик помилок на ранніх етапах роботи.

Моделі системи FSM та DAG діють також у зв'язку, доповнюючи одна одну. Наприклад, коли рішення u переходить у стан *Superseded* або *Archived* у своєму життєвому циклі, система автоматично запускає каскадний аналіз впливу $Impact(u)$ та генерує подію *trigger_review* для всіх рішень, пов'язаних з u ребром *basis*.

2.3 Побудова ієрархічної структури робіт (WBS)

Перехід від математичного моделювання до фактичної розробки системи «DECY» потребує чіткої декомпозиції завдань. Цю задачу вирішує ієрархічна структура робіт WBS – фундаментальний інструмент проєктного управління, який перетворює концептуальні вимоги на конкретні кроки для команди. WBS розбиває проєкт на керовані етапи та виступає єдиною базою для подальшого календарного планування, розрахунку вартості та контролю виконання.

Для управління проєктом розробки інформаційної системи «DECY» застосовано метод декомпозиції, орієнтованої на результати, відповідно до стандартів PMBoK. Замість переліку операційних процесів, таких як написання коду, налаштування серверів, тощо, структура фокусується на постачанні готових архітектурних компонентів: ядро управління життєвим циклом рішень, клієнтський інтерфейс, модулі інтеграції з Jira, тощо. Відмова від класичної процесної структури зумовлена вимогами ітеративної розробки. Модель, орієнтована на результати, гарантує, що кожен вузол WBS виступає як самостійний інкремент продукту, який має чіткі критерії приймання і може бути виконаним у межах одного циклу розробки [11].

Побудова WBS проєкту «DECY» спирається на два базові принципи:

– ***«Правило 100%»*** гарантує повноту обсягу. Сума підзадач має точно дорівнювати обсягу батьківського вузла без прогалин і зайвої роботи;
– ***Правило «8/80»***, яке обмежує розмір базового пакета робіт. На практиці це означає декомпозицію до рівня завдань тривалістю 1–2 інженерні тижні (8–80 годин), що призначаються одному виконавцю. Такий розмір забезпечує прозорість прогресу і дозволяє уникнути надмірного мікроменеджменту.

Структуру робіт верхніх рівів в графічному вигляді наведено на рис. 2.4. На першому рівні проєкт декомпозовано на сім гілок, що відповідають семи структурним напрямкам розробки.

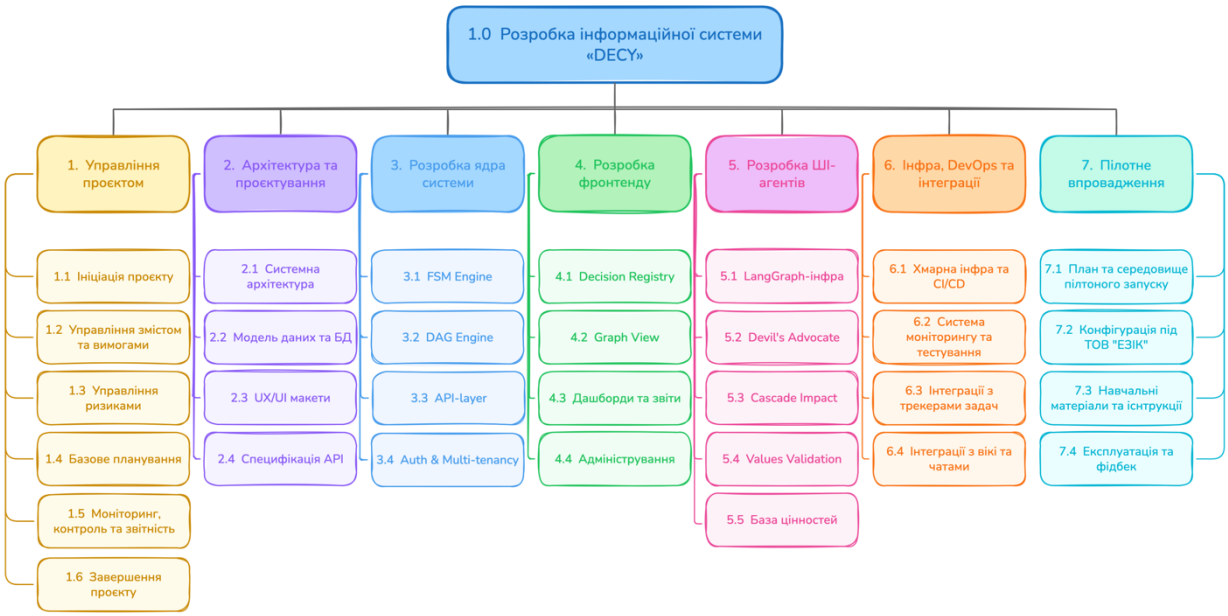

Рис. 2.4. WBS верхніх рівнів

Архітектура верхнього рівня WBS (рис. 2.4) поділена за ключовими компетенціями для забезпечення паралельної розробки де це можливо. Створення ядра та клієнтської частини ведуться незалежно на основі API-контракту, а ШІ-агенти виділені в окрему гілку через специфіку стеку LangChain. Гілку «Управління проєктом» включено до структури як повноцінний пакет робіт для генерації звітності та коректного обліку управлінських трудовитрат у системі EVM. При цьому розробка технічної

документації не виокремлюється, вона постачається разом із відповідним програмним компонентом як невід'ємна частина критеріїв готовності (Definition of Done).

Розглянемо деталізацію робіт до третього рівня у роботі з управління проєктом у табл. 2.4.

Таблиця 2.4

Деталізована WBS по управлінню проєктом

Код	Робочий пакет
1	**Управління проєктом**
1.1	Ініціація проєкту
1.1.1	*Розробка паспорту (уставу) проєкту*
1.1.2	*Формування реєстру стейкхолдерів*
1.1.3	*Розробка бізнес-кейсу та фінансового обґрунтування*
1.2	Управління змістом та вимогами
1.2.1	*Формування та пріоритизація беклогу продукту*
1.2.2	*Розробка специфікації вимог (SRS)*
1.2.3	*Створення матриці трасування вимог (RTM)*
1.3	Управління ризиками
1.3.1	*Ідентифікація та формування реєстру ризиків*
1.3.2	*Побудова матриці ймовірності та наслідків*
1.3.3	*Розробка плану реагування на ризики*
1.4	Базове планування
1.4.1	*Затвердження базового розкладу*
1.4.2	*Затвердження базового кошторису*
1.4.3	*Розробка плану управління комунікаціями*
1.5	Моніторинг, контроль та звітність
1.5.1	*Формування статус-звітів за методом освоєного обсягу*
1.5.2	*Ведення журналу запитів на зміни*
1.5.3	*Ведення журналу проблем*
1.6	Завершення проєкту та передача в експлуатацію
1.6.1	*Розробка документа засвоєних уроків*
1.6.2	*Підготовка акту передачі системи в операційну діяльність*
1.6.3	*Підготовка підсумкового звіту про виконання проєкту*

Як видно з наведеної таблиці, гілка управління проєктом декомпозована за класичним життєвим циклом (від ініціації до закриття) і повністю відповідає результато-орієнтованому підходу. Кожен робочий пакет нижнього рівня це конкретний управлінський артефакт (документ, базовий план або звіт). Це

створює надійний адміністративний каркас, який гарантує прозорість контролю на всіх етапах розробки платформи «DECY» та забезпечує її безшовну інтеграцію в операційну діяльність компанії після завершення проєкту.

Розглянемо також деталізацію наступного кластеру робіт у табл. 2.5 – архітектура та проєктування.

Таблиця 2.5

Деталізована WBS по архітектурі та проєктуванню

Код	Робочий пакет
2	**Архітектура та проєктування**
2.1	Системна архітектура
2.1.1	*Розробка високорівневої архітектурної схеми HLD*
2.1.2	*Створення UML-подібних діаграм взаємодії компонентів системи*
2.1.3	*Формування вимог до інфраструктури та безпеки*
2.2	Модель даних та БД
2.2.1	Розробка логічної та фізичної ER-моделі реляційної БД
2.2.2	Проєктування структури графових зв’язків для пітримки DAG
2.2.3	Налаштування процесів міграцій БД та конфігурація ORM
2.3	UX/UI-макети та дизайн-система
2.3.1	*Створення корпоративної дизайн-системи (UI-kit)*
2.3.2	*Розробка інтерактивних прототипів ключових користувацьких сценаріїв*
2.3.3	*Затвердження фінальних макетів екранів*
2.4	Специфікація API
2.4.1	*Розробка контракту взаємодії OpenAPI*
2.4.2	*Проєктування GraphQL-схем для обробки графових запитів*
2.4.3	*Проєктування архітектури інтеграцій з third-party systems (Jira, Notion, Confluence, тощо) через REST API та MCP*

Архітектурна гілка закладає фундамент, на якому згодом буде вестись робота у гілках 3–6. Її артефактами є не код, а специфікації, схеми та контракти. Послідовність пакетів вибудувана як рух від загального до часткового: спочатку системна архітектура (2.1) задає рамку, далі в ній деталізуються модель даних (2.2) та користувацькі сценарії (2.3), а специфікація API (2.4) фіксує контракт між бекендом і фронтендом, даючи

можливість далі розвиватися паралельно. За схожим принципом робиться декомпозиція інших гілок.

2.4 Ресурсно-вартісне планування проєкту

Перехід від WBS до повноцінного плану проєкту потребує трьох послідовних кроків: визначення команди та ролей, оцінки трудовитрат та формування кошторису.

2.4.1 Команда проєкту та ролі

При формуванні команди розробки «DECY» враховуються наступні ключові фактори:

- Специфіка технологічного стеку інформаційної системи
- Таймінг до запуску пілоту – 6 місяців
- Бюджетні реалії невеликого IT-бізнесу в Україні, зокрема ТОВ «ЕЗІК»

В основу ресурсної стратегії покладено принцип крос-функціональності та раціонального суміщення компетенцій. Наприклад, DevOps компетенції делеговано технічному ліду (Tech Lead / Solution Architect). Оскільки архітектура «DECY» розгортається в межах єдиного хмарного середовища GCP, залучення виділеного DevOps-фахівця було б економічно недоцільним через неминучі простої ресурсу. До того ж, ТОВ «ЕЗІК» вже має перевіреного фахівця на цю роль з усіма необхідними компетенціями та досвідом роботи. UX/UI-дизайнер залучається на 0,5 FTE з основним завантаженням у перші місяці (формування дизайн-системи та прототипів) і подальшою підтримкою за вимогою.

Для економії бюджету деякі ролі будуть залучені лише на другому кварталі проєкту (останні 3 міс.), а саме QA та AI/ML фахівці. Це рішення спирається на природну послідовність робіт. QA-інженер ефективно працює лише за наявності стабільної функціональності для систематичного тестування, тому на ранніх етапах якість підтримується самими розробниками.

AI/ML-інженер реалізує гілку 5 WBS (інтелектуальні агенти на LangGraph), яка архітектурно надбудована поверх готових FSM та DAG реалізацій – без них агенти не мають з чим взаємодіяти. Залучення AI/ML-фахівця з першого дня призвело б до простою ресурсу впродовж перших трьох місяців, тоді як зміщення його входу в проєкт на Q2 синхронізує готовність API ядра системи з початком роботи над агентами.

Деталізований склад команди з годинними ставками наведено в табл. 2.6. Ставки свідомо трохи вищі за середні ринкові показники для українського IT-ринку 2026 задля залучення найталановитіших спеціалістів.

Таблиця 2.6

Склад команди проєкту розробки «DECY»

Роль	FTE	Період залучення	Тривалість (міс.)	Ставка (USD/год.)	Людино-години
Project Manager	0,5	Q1-Q2	6	35	504
Tech Lead / Solution Architect	1,0	Q1-Q2	6	50	1008
Backend Engineer	1,0	Q1-Q2	6	40	1008
Frontend Engineer	1,0	Q1-Q2	6	40	1008
UX/UI Designer	0,5	Q1-Q2	6	30	504
AI/ML Engineer	1,0	Q2	3	40	504
QA Engineer	1,0	Q2	3	30	504
Разом / TOTAL	—	—	—	—	5040

При розрахунку використано виробничий календар із 21 робочим днем на місяць та 8 робочими годинами на день, що дає базу 168 годин на місяць для 1,0 FTE [24]. Сумарний фонд робочого часу команди становить 5 040 людино-годин (близько 630 людино-днів).

2.4.2 Оцінка трудовитрат за пакетами WBS

Сумарний фонд робочого часу команди в обсязі 5 040 людино-годин (табл. 2.6) є валовим показником, але цей показник ще нічого не говорить про реалістичність плану. Для переходу до повноцінного календарного

планування та формування кошторису потрібен розподіл цього фонду між сімома гілками WBS з урахуванням того, які саме ролі залучені до робіт по кожній з них. Оцінку виконано за методом «bottom-up». Зведений розподіл наведено в табл. 2.7.

Таблиця 2.7

Розподіл трудовитрат за гілками WBS у розрізі ролей, людино-години

Гілка WBS	PM	TL/ SA	BE	FE	UI/ UX	AI/ ML	QA	Разом	Частка
1. Управління проєктом	504	—	—	—	—	—	—	—	10,0%
2. Архітектура та проєктування	—	280	200	50	254	—	—	784	15,6%
3. Розробка ядра системи	—	300	700	—	—	—	180	1180	23,4%
4. Розробка фронтенду	—	—	—	700	250	—	100	1050	20,8%
5. Розробка ШІ-агентів	—	130	50	—	—	450	100	730	14,5%
6. Інфра, DevOps, інтеграції	—	250	50	200	—	—	50	550	10,9%
7. Пілотне впровадження	—	48	8	58	—	54	74	242	4,8%
Разом	504	1008	1008	1008	504	504	504	5040	100%

Структура трудовитрат відображає як технологічний профіль системи, так і обмеження команди. Найважчою гілкою очікувано виявилося ядро системи (23,4% фонду), саме тут реалізуються математичні моделі і саме сюди скеровано 70% часу Backend Engineer. Друга за вагою гілка – фронтенд (20,8%), в якій реалізовані інтерактивна візуалізація DAG-графу та інтерфейс Decision Registry. Разом ці дві гілки забирають 44% усього фонду, що відповідає галузевому орієнтиру для B2B SaaS-продуктів аналогічної складності.

Гілка 7, яка відповідає за пілотне впровадження, свідомо отримала найменшу частку – 4,8%. Це не недооцінка важливості пілоту, а наслідок того,

що значна частина роботи з адаптації під ТОВ «ЕЗІК» (наприклад, налаштування шаблону життєвого циклу або заповнення бази корпоративних цінностей) виконується вже існуючою командою «ЕЗІК» поза рамками цього проєкту і тому не входить у зведений фонд.

2.4.3 Календарний план та критичний шлях

Розподіл трудовитрат у поєднанні зі складом дозволяє побудувати календарний план проєкту. Для цього необхідно встановити логічні залежності між гілками WBS та визначити критичний шлях. Критичний шлях – найдовша послідовність робіт, скорочення якої прямо скорочує тривалість проєкту в цілому. Графічне представлення плану у вигляді діаграми Ґанта наведено на рис. 2.5.

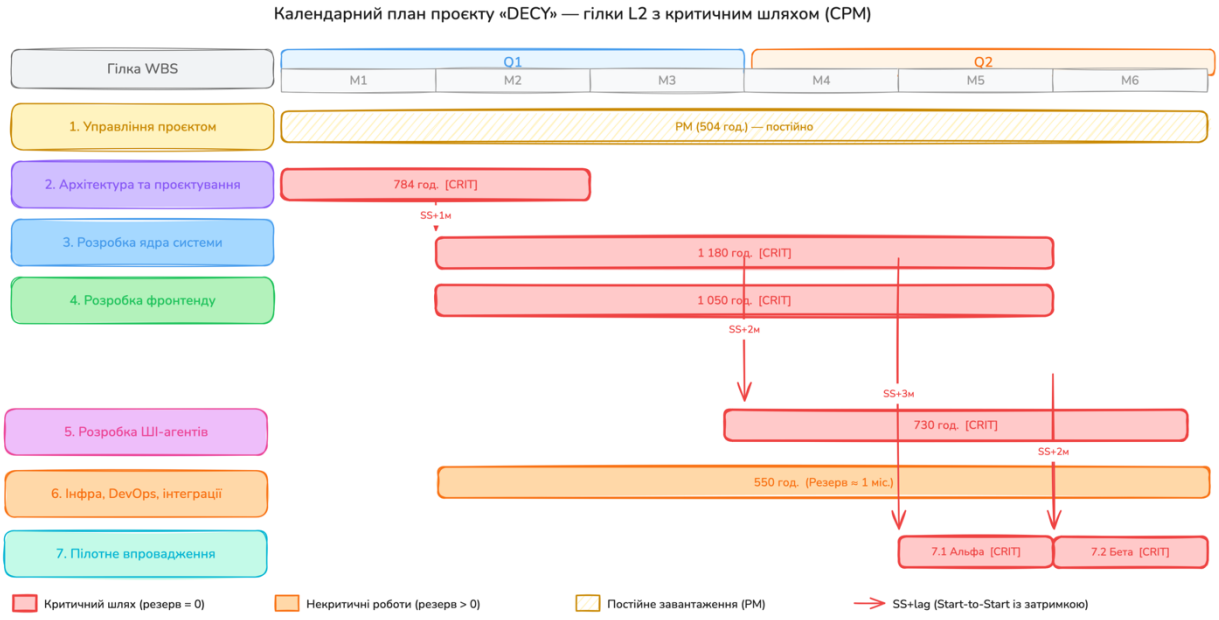

Рис. 2.5. Календарний план і критичний шлях

Для моделювання було використано програмне забезпечення GanttProject 3.4 на операційній системі MacOS. Фрагмент діаграми Ґанта у інтерфейсі програмного забезпечення наведено на рис 2.6.

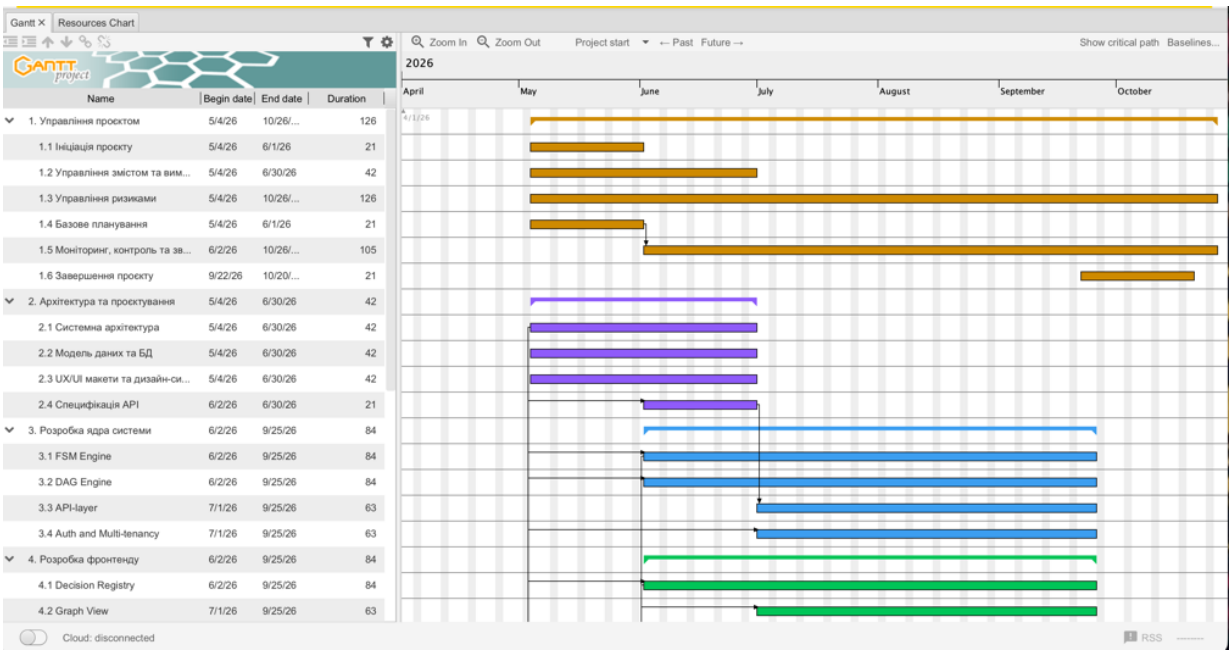

Рис. 2.6. Фрагмент діаграми Ґанта у GanttProject

Для скорочення загальної тривалості розробки до цільових 6 місяців застосовано метод стиснення розкладу Fast Tracking, який робить можливим паралельне виконання фаз. Відповідно до цього підходу, залежності між ключовими пакетами робіт змодельовано за типом *Старт-Старт із затримкою* (SS+lag) замість FS (фініш-старт).Архітектурна гілка не повинна бути завершена повністю для початку реалізації. Як тільки зафіксована референс-архітектура та схема API (приблизно через місяць після старту), бекенд, фронтенд та DevOps спеціалісти можуть розпочати роботу паралельно. Аналогічно, ШІ-агенти стартують через два місяці після початку розробки ядра, коли публічний API вже стабільний для інтеграції. Зведення залежностей наведено в табл. 2.8.

Таблиця 2.8

Логічні залежності між гілками WBS

Залежніть	Тип	Лаг	Зміст
1	2	3	4
2 → 3	SS	+ 1міс.	Ядро стартує після фіксації архітектури та API-контракту
2 → 4	SS	+ 1міс.	Фронтенд стартує після затвердження дизайн-системи та прототипів

Продовження таблиці 2.8

1	2	3	4
2 → 6	SS	+ 1міс.	Інфра стартує після фіксації цільового стеку
3 → 5	SS	+ 2міс.	Агенти стартують після появи стабільного публічного API
3 → 7.1	SS	+ 3міс.	Альфа-пілот стартує при готовності ядра ~75%
4 → 7.1	SS	+ 3міс.	Альфа-пілот стартує при готовності фронтенду ~75%
5 → 7.2	SS	+ 2міс.	Бета-пілот стартує при готовності агентів ~67%
7.1 → 7.2	FS	—	Бета-пілот стартує після завершення Альфа-пілоту

Для забезпечення безперервної валідації продукту, сам пілотний запуск розділено на дві послідовні стадії. Альфа-пілот у М5 проходить на функціональному ядрі без ШІ-агентів. Користувачі ТОВ «ЕЗІК» працюють з Decision Registry та Graph View, перевіряючи базовий цикл фіксації рішень, типізацію залежностей та каскадний аналіз впливу. Бета-пілот у М6 починається після того, як ШІ-агенти виходять на стадію інтеграційної готовності. Така двофазна структура дозволяє зібрати валідаційні дані щодо базової функціональності ще до завершення розробки інноваційної ШІ-надбудови та забезпечує ізольоване тестування ризикованої частини продукту, в якій найбільше непевностей із точки зору експлуатаційної стабільності.

Розрахунок критичного шляху виконано стандартною процедурою прямого та зворотного проходу (forward/backward pass). Особливістю отриманої сітки є розгалужений критичний шлях. На ньому одночасно лежать дві паралельні гілки робіт, кожна з яких упирається в кінцеву віху проєкту:

- ***Шлях А*** – Архітектура → Ядро → Альфа-пілот → Бета-пілот (2 → 3 → 7.1 → 7.2);
- ***Шлях В*** – Архітектура → Ядро → ШІ-агенти → Бета-пілот (2 → 3 → 5 → 7.2).

Шлях С через гілку 4 (Frontend) має резерв часу близько 0,5 місяця і кваліфікується як sub-critical, тобто формально не критичний, але настільки близький до критичного, що при моніторингу потребує такої ж пильної уваги, як і червоні гілки. Помітний резерв часу має лише гілка 6 (Інфра, DevOps, інтеграції) у розмірі 1 місяця, що й виправдовує її подовжену тривалість (M2–M6) при відносно невеликих трудовитратах. Цей резерв дозволяє гнучко балансувати завантаження TL/SA, який одночасно відповідає за DevOps-роботи цієї гілки та за архітектурний нагляд на критичних гілках.

Розгалуженість критичного шляху накладає підвищені вимоги на дисципліну виконання. Затримка в будь-якій із трьох технічних гілок (3, 4, або 5) безпосередньо переноситься у фінальну дату завершення проєкту. Особливо чутливими є дві віхи:

1. ***Перехід M2/M3 (Handoff архітектури).*** Момент, коли архітектурна команда передає результати для подальшої реалізації. Запізнення з фіналізацією API-контракту або моделі даних блокує одночасно три критичні гілки.
2. ***Перехід M4 (Старт ШІ-розробки).*** Затримка зі стартом гілки 5 не може бути компенсована нарощуванням ресурсу, оскільки роль зайнята одним фахівцем у фіксованому часовому вікні.

2.4.4 Кошторис проєкту

Розрахунок кошторису виконано за методом «bottom-up», тобто знизу вгору. Спершу обчислено базові статті прямих витрат, далі сформовано резерви на ризики згідно з рекомендаціями PMI, і лише на останньому кроці отримано загальний бюджет проєкту. Кошторис складено в доларах США, оскільки ставки команди та частина зовнішніх сервісів номіновані саме в цій валюті, а кінцевий продукт орієнтований на B2B-ринок з ціноутворенням у USD.

До кошторису включено лише ті витрати, які виникають безпосередньо у зв’язку з проєктом «DECY». Загальні адміністративно-операційні витрати

ТОВ «ЕЗІК», такі як бухгалтерсько-кадровий супровід, оренда офісу, корпоративні підписки, тощо – існують незалежно від проєкту і тому з кошторису свідомо виключені.

Структуру кошторису сформовано з трьох логічних блоків. Фонд оплати праці є найбільшою статтею (~86%) і обчислюється як добуток годинних ставок (табл. 2.6) на трудовитрати кожної ролі. Інфраструктурні та інструментальні витрати включають хмарну інфраструктуру, LLM-API для розробки та тестування ШІ-агентів, а також підписки на інструменти розробки (GitHub, Linear, моніторинг, дизайн-інструменти). Юридичні витрати на цьому етапі сфокусовані на одній основній статті – розробці пакету GIG-контрактів для залучення команди в режимі резидента Дія Сіті, включно з умовами IP-assignment, що забезпечують консолідацію прав інтелектуальної власності на ТОВ «ЕЗІК».

Управлінський резерв в обсязі 5% від базового кошторису закладено для невідомих ризиків, які не були ідентифіковані на етапі планування і можуть з’явитися під час виконання. Цей резерв перебуває поза базовим кошторисом, контролюється спонсором проєкту й активується лише за рішенням комітету змін.

Сукупний рівень резервів (9,3% від загального бюджету) відповідає рекомендованому діапазону для проєктів середньої складності та забезпечує адекватний буфер на відомі та невідомі ризики.

Зведений кошторис проєкту «DECY» з розрахованим загальним бюджетом у розмірі 233 600 USD наведено в табл. 2.9.

Таблиця 2.9

Зведений кошторис проєкту «DECY»

№	Стаття витрат	Сума, USD	Частка, %
1	**2**	**3**	**4**
1.	Прямі витрати		
1.1	Фонд оплати праці команди	201 600	86,3%
1.2	Хмарна інфраструктура (GCP)	4 800	2,0%

Продовження таблиці 2.9

1	2	3	4
1.3	LLM-API для розробки та тестування ШІ-агентів	3000	1,3%
1.4	Інструменти розробки	1000	0,4%
1.5	Юридичні витрати – GIG-контракти Дія Сіті, IP-assignment	1500	0,7%
—	Підсумок прямих витрат	211 900	90,7%
2.	Резерви		
2.1	Резерв непередбачених витрат (5% від п.1)	10 600	4,5%
2.2	Управлінський резерв (5% від базового кошторису)	11 100	4,8%
—	Підсумок резервів	21 700	9,3%
—	**Загальний бюджет проєкту**	**233 600**	**100,0%**

2.4.5 Проєктування системи контролю виконання на основі методу освоєного обсягу

Метод освоєного обсягу об'єднує три виміри проєктної динаміки: обсяг, час та вартість у єдину аналітичну рамку [13]. EVM застосовується у проєкті «DECY» як базовий механізм оперативного управління виконанням.

В основі методу лежать три первинні показники, що обчислюються на кожну звітну дату:

- ***Planned Value (PV)*** – планова вартість робіт, які повинні бути виконані до звітної дати згідно з кошторисом. Для проєкту «DECY» розраховується як кумулятивна сума запланованих витрат за пакетами WBS відповідно до календарного плану (рис. 2.5);
- ***Earned Value (EV)*** – освоєний обсяг, тобто планова вартість робіт, які фактично завершені до звітної дати. Якщо станом на М3 завершено 50% пакета 3.1 (FSM Engine) із плановим бюджетом, скажімо, $20 000, то EV цього пакета становить $10 000, незалежно від того, скільки команда фактично витратила на його виконання;
- ***Actual Cost (AC)*** – фактично понесені витрати на виконання робіт до звітної дати, отримані з облікової системи.

З цих трьох первинних показників обчислюються чотири похідні метрики, які власне й роблять EVM управлінським інструментом:

- $SV = EV - PV$ (Відхилення за графіком);
- $CV = EV - AC$ (Відхилення за вартістю);
- $SPI = EV/PV$ (Індекс виконання за графіком);
- $CPI = EV/AC$ (Індекс виконання за вартістю).

Значення SPI або CPI ≥ 1 свідчать про виконання за планом або з випередженням, значення < 1 про відставання чи перевитрати. На відміну від абсолютних SV/CV, індекси є безрозмірними і дозволяють порівнювати стан різних пакетів робіт та проєктів між собою.

Окрім діагностики поточного стану, EVM дає змогу обчислювати прогноз кінцевих показників проєкту на основі вже накопиченої статистики виконання. $EAC = BAC/CPI$, де EAC – Estimate at Completion (прогноз вартості завершення), BAC – Budget at Completion (загальний базовий кошторис проєкту, без урахування управлінського резерву). Якщо CPI на середині проєкту становить 0,85, то прогнозна вартість завершення вже становитиме $\$222\,500 / 0{,}85 \approx \$261\,800$, що сигналізує керівнику проєкту про необхідність втручання задовго до того, як прямі фактичні витрати перевищать бюджет.

Для забезпечення дієвого контролю за проєктом алгоритм використання EVM адаптовано до потреб гнучкої розробки. Звітний період синхронізовано з двотижневими ітераціями розробки проєкту (спринтами) команди.

З метою мінімізації адміністративного навантаження на керівника проєкту, від суб'єктивного методу оцінки «відсотка виконання» було вирішено відмовитися. Натомість застосовується комбінований підхід до нарахування освоєного обсягу (EV):

- ***Метод «0/100»*** – для робочих пакетів тривалістю до одного спринта. Обсяг зараховується виключно після повного закриття задачі за критеріями Definition of Done;

- ***Метод «50/50»*** – для пакетів, що тривають довше двох тижнів. 50% вартості зараховується в момент старту робіт, інші 50% по їх завершенню.

Для оперативного управління відхиленнями встановлено чіткі межі ескалації:

- Зниження індексів ***CPI або SPI < 0,9*** (жовта зона) ініціює внутрішній аналіз причин та розробку коригувальних дій на рівні команди;
- Падіння показників до ***CPI або SPI < 0,8*** (червона зона) вважається критичним. Це вимагає негайної ескалації проблеми на рівень спонсора проєкту для можливого залучення управлінського резерву бюджету.

Впровадження EVM у такому форматі дає проєкту надійний інструмент раннього попередження замість простої фіксації збитків. Команда отримує життєво необхідний запас часу, щоб вирівняти ситуацію, щойно показники починають просідати, а не коли дедлайн уже минув. Усі необхідні роботи зі збору та аналізу цих даних уже врахована в загальному графіку як пакет робіт 1.5 «Моніторинг, контроль та звітність».

РОЗДІЛ 3. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «DECY»

Проєктування архітектури є фундаментальним етапом створення інформаційної системи «DECY», що забезпечує її надійність, масштабованість та практичну реалізацію закладених математичних моделей. У цьому розділі розроблено концептуальну архітектуру, підібраний технологічний стек, спроєктована структура бази даних реєстру рішень та механізми графової візуалізації, розглянута архітектура ШІ-агентів та логіка користувацького інтерфейсу.

3.1 Концептуальна архітектура системи та вибір технологічного стеку

У сучасній практиці B2B SaaS-розробки зазвичай використовують одну з трьох моделей:

- ***Моноліт*** – цілісна система зі спільною кодовою базою, де всі компоненти тісно взаємопов’язані;
- ***Модульний моноліт*** – єдиний програмний комплекс із чітко відокремленими внутрішніми модулями та контрактами між ними;
- ***Мікросервісна архітектура*** – набір автономних компонентів, що взаємодіють через мережеві протоколи.

Для проєкту «DECY» обрано модульний моноліт як основну архітектурну модель. Цей вибір є оптимальним компромісом між поточними обмеженнями проєкту та планами на майбутнє. Для команди з 5–6 інженерів мікросервісна архітектура створить забагато інфраструктурної рутини, яка не принесе відповідної користі. На стадії пілотного запуску вимоги до системи ще можуть змінюватися, і переписувати та адаптувати модульний моноліт набагато дешевше, ніж вносити зміни в мережу незалежних сервісів. Такий підхід також суттєво скорочує час виходу на ринок.

Окреме виключення зроблено для сервісу ШІ-агентів. Так як це принципово інша експертиза, і над кластером робіт по ній буде працювати окремий інженер (а в подальшому окрема команда), було прийнято рішення виокремити сервіс ШІ-агентів у окремий компонент інформаційної системи – із своєю відокремленою кодовою базою та контейнеризацією.

Вибір технологічного стеку для «DECY» здійснено принципом TypeScript end-to-end. Один і той самий мовний контракт використовується від моделі бази даних до користувацького інтерфейсу, що дає три суттєві переваги: спільні типи даних між бекендом та фронтендом – доменні сутності описуються один раз і використовуються в обох рівнях, спрощений найм (один і той самий профіль інженерів закриває обидва рівні стека) та зниження когнітивного навантаження при роботі команди над різними модулями. Деталізований стек із обґрунтуванням вибору наведено в табл. 3.1.

Таблиця 3.1

Технологічний стек «DECY»

Компонент	Технологія	Обґрунтування вибору
1	2	3
Backend Runtime	Node.js + NestJS	Модульна архітектура, dependency injection, природньо підтримує модульний моноліт із чіткими межами модулів ПЗ
Programming Language	TypeScript	Type safety, спільні типи між різними компонентами ПЗ, стандарт індустрії для сучасних B2B SaaS
ORM	Prisma	Автогенерація type-safe моделей з SQL-схеми; простота міграцій; нативна інтеграція з NestJS і TypeScript
Database	PostgreSQL (NeonDB)	Реляційна модель гарно підходить під заплановану схему даних, різноманітний набір extensions для додаткового функціоналу (наприклад роботи за графами), стандарт індустрії для сучасних B2B SaaS
Multi-tenancy	NeonDB per-tenant provisioning	Ізоляція даних окремих компаній на рівні окремих БД (database-per-tenant) – необхідна для B2B-продукту з чутливими даними про управлінські рішення. Автоматичне severless масштабування.

Продовження таблиці 3.1

1	2	3
Frontend	React + TypeScript + Vercel	Зріла екосистема, найм, нативна інтеграція з типами бекенду, Vercel – гарний DX
Graph Visualization	React Flow + dagre	Сучасний DX, інтеграція з React-стеком, dagre для hierarchical layout DAG-структури
Agentic Framework	LangChain + LangGraph	Забезпечує детермінований контроль потоку виконання (FSM) та збереження станів агентів для тривалих асинхронних процесів
LLM Provider	Anthropic Claude + Google Gemini	Велике контекстне вікно, безпека даних, якість моделей
Cloud Provider	Google Cloud Platform	Наявний досвід команди, великий набір сервісів для ШІ

Серед наведених у табл. 3.1 архітектурних рішень окремого фокусу потребує підхід до організації ізоляції даних (multi-tenancy) на базі NeonDB.

Оскільки «DECY» є B2B-продуктом, який оперує найбільш чутливою інформацією компаній-клієнтів (стратегічні плани, архітектурні компроміси, рішення щодо персоналу чи ціноутворення), фундаментальною вимогою до системи є абсолютна ізоляція контекстів. Традиційна для багатьох SaaS-проєктів модель shared-schema, коли всі клієнти знаходяться в одній базі даних і розділяються лише полем tenant_id була свідомо відхилена. Хоча вона дешевша у впровадженні та менеджменті, така модель несе неприпустимий ризик витоку даних у разі програмних помилок і унеможливлює надання B2B-замовникам жорстких гарантій безпеки на рівні контрактних SLA.

Натомість для платформи обрано архітектуру *database-per-tenant* – фізично окрема база даних для кожної компанії-клієнта. Використання хмарної СУБД NeonDB робить цей підхід економічно та операційно доцільним завдяки serverless-моделі тарифікації. Обчислювальні ресурси споживаються та масштабуються лише в моменти активності конкретної компанії, зводячи вартість простою до нуля. Додатковою перевагою цього рішення є механізм *database branching*, який дозволяє миттєво створювати ізольовані копії баз даних для безпечного тестування без жодного ризику для production-

середовища клієнтів. Окрім ізоляції безпеки, така архітектура водночас виконує роль масштабування завдяки механізму шардінгу – навантаження одного активного tenant'a не впливає на продуктивність інших, а додавання нової компанії реалізується автоматизованим NeonDB per-tenant provisioning і не потребує реконфігурації кластера.

Архітектура інформаційної системи «DECY» у нотації C4 наведена на рис. 3.1.

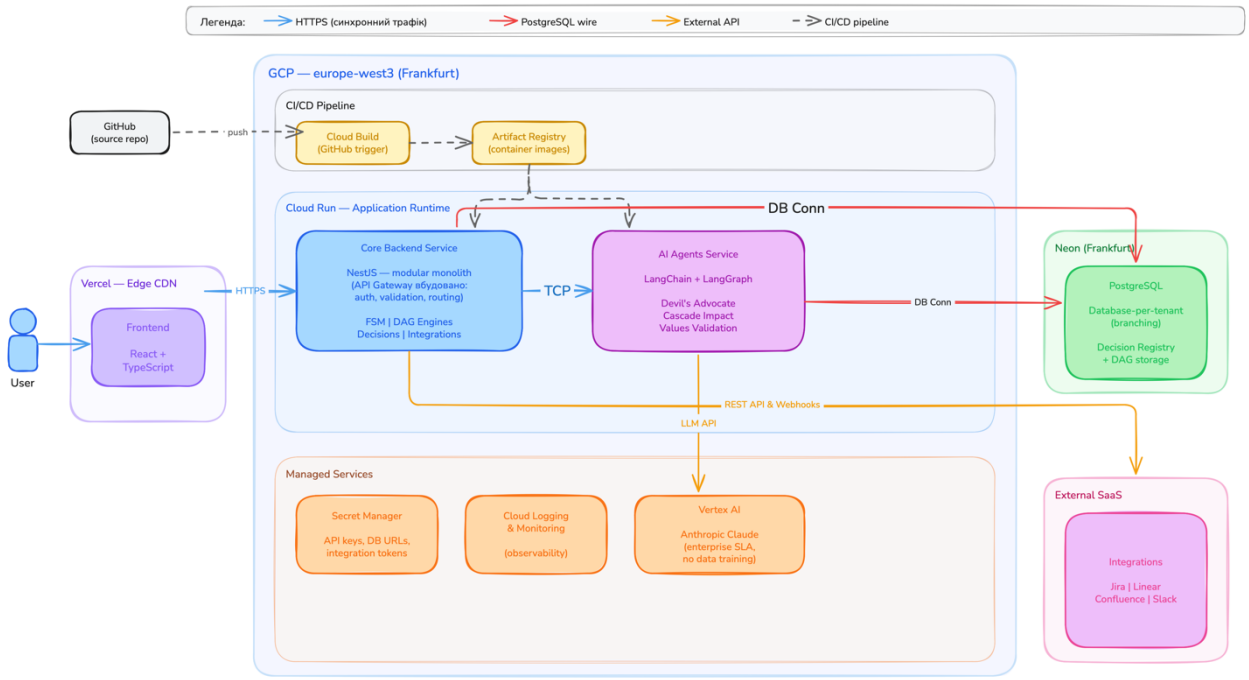

Рис. 3.1. C4 Архітектура «DECY»

Архітектура платформи розгорнута в екосистемі Google Cloud Platform (GCP) і складається з трьох рівнів:

- ***CI/CD pipeline*** – використовує Cloud Build та Artifact Registry для автоматичної доставки нових версій коду з GitHub у середовище Cloud Run;
- ***Runtime*** – містить основний бекенд, який розгортається за допомогою Cloud Run. Це модульний моноліт на NestJS, що обробляє бізнес-логіку і виокремлений сервіс ШІ-агентів на базі LangGraph;

– ***Managed Services*** – включають Vertex AI для роботи з мовними моделями, Secret Manager для безпечного зберігання ключів та Cloud Logging для моніторингу системи.

Фронтенд винесено на платформу Vercel для швидкої географічно розподіленої доставки клієнтської частини. База даних PostgreSQL розгорнута в хмарному сервісі Neon фізично в тому ж регіоні (Франкфурт), що й основний кластер GCP. Це гарантує мінімальні затримки мережі та спрощує дотримання європейських норм захисту даних на кшталт GDPR. Зовнішні робочі сервіси (Jira, Linear, Confluence, Slack) виділені в окремий блок, з якими «DECY» обмінюється даними через REST API та Webhooks.

На етапі пілотного впровадження для ТОВ «ЕЗІК» система свідомо розгортається лише в одному регіоні. Побудова складної мультирегіональної інфраструктури на цьому етапі створила б зайве технічне навантаження та подовжило би строки виконання проєкту.

3.1.1 Архітектура інформаційної безпеки та захисту даних

Як вже було зазначено вище, оскільки «DECY» працює з дуже чутливим типом корпоративної інформації – управлінськими рішеннями, ціноутворення, технологічні компроміси, кадрові питання клієнтів, тощо, питання інформаційної безпеки розглядається в проєкті як архітектурне обмеження, що безпосередньо впливає на вибір технологічного стеку та модель розгортання. Захист побудовано за принципом ешелонованої оборони на чотирьох рівнях: ізоляції даних, шифрування, контролю доступу та особливого режиму роботи із зовнішніми LLM-сервісами.

Базовою лінією захисту є архітектура *database-per-tenant*, описана вище: кожна компанія-клієнт отримує фізично окрему базу даних, що структурно унеможливлює сценарій витоку даних чужої компанії через помилку в SQL-фільтрації. Внутрішня комунікація між сервісами в межах Cloud Run відбувається в приватному мережевому периметрі; жоден сервіс бекенду не має публічної експозиції, крім явно заданих точок входу (див. рис. 3.1).

Дані захищаються шифруванням на двох рівнях:

- ***In transit*** – увесь зовнішній і внутрішній трафік передається через TLS 1.3. Клієнт → бекенд по HTTPS, бекенд ↔ БД через шифроване з'єднання PostgreSQL із обов'язковим режимом *sslmode*, бекенд ↔ LLM-провайдер по HTTPS у межах Vertex AI;
- ***At rest*** – сховище NeonDB використовує шифрування на рівні дисків стандартом AES-256, що поширюється також на резервні копії. Усі програмні секрети (токени API, ключі підключення до баз даних, інтеграційні токени до зовнішніх SaaS-систем) зберігаються виключно в GCP Secret Manager і не присутні в коді чи змінних середовища у відкритому вигляді.

Автентифікація користувачів реалізована через JWT-токени з refresh-механізмом. Це стандартна практика для екосистеми NestJS. Кожен запит до API проходить *tenant-scoping* – система верифікує, що ідентифікатор tenant у токені відповідає підключенню до бази даних конкретної компанії, що блокує спроби доступу до чужих компаній навіть у випадку компрометації токена. У межах кожної окремої компанії передбачено рольову модель з диференціацією прав за типом операції – читання, редагування та адміністрування. Усі операції зі змінами стану рішень або їхніх атрибутів фіксуються в журналі аудиту з обов'язковим записом ідентифікатора користувача, типу операції та часової мітки.

Рівень роботи з зовнішніми LLM-сервісами потребує особливої уваги, оскільки контекст управлінського рішення, що передається до моделі, може містити чутливу інформацію:

- ***Провайдер із гарантією no-training*** – як основний LLM-провайдер обрано Anthropic Claude через Google Vertex AI, який надає контрактну гарантію enterprise SLA, що вхідні дані не використовуються для тренування моделей та не зберігаються поза часом обробки запиту на відміну від публічних API LLM-провайдерів;

- ***Ізоляція трафіку в межах GCP*** – Vertex AI розгорнуто в тому ж регіоні, що й сервіси «DECY», тому запити до LLM не виходять за межі інфраструктури Google Cloud і не передаються по публічному інтернету;
- ***Мінімізація контексту –*** перед формуванням запиту атрибути рішення пропускаються через алгоритм маскування, який замінює поля, помічені конфігурацією компанії як чутливі (наприклад, цифри ціноутворення чи імена контрагентів), на замасковані значення («***», тощо).

3.2 Проєктування логічної моделі бази даних

Логічна модель бази даних «DECY» спроєктована навколо центральної доменної сутності – Рішення (Decision). Вся архітектура даних розділена на чотири функціональні блоки, що забезпечують повний життєвий цикл прийняття рішень:

- Користувачі та проєктний контекст – визначають приналежність рішення та рольову модель доступу;
- Модель залежностей – реалізує математичну DAG-модель, де кожне ребро має чітку типізацію;
- Система цінностей – забезпечує зв’язок рішень із корпоративними цінностями та результатами перевірок ШІ-агентами;
- Аудиторські журнали – фіксують історію змін станів та активність інтелектуальних агентів.

Повноцінна ER-діаграма спроєктованої бази даних наведена на рис. 3.2.

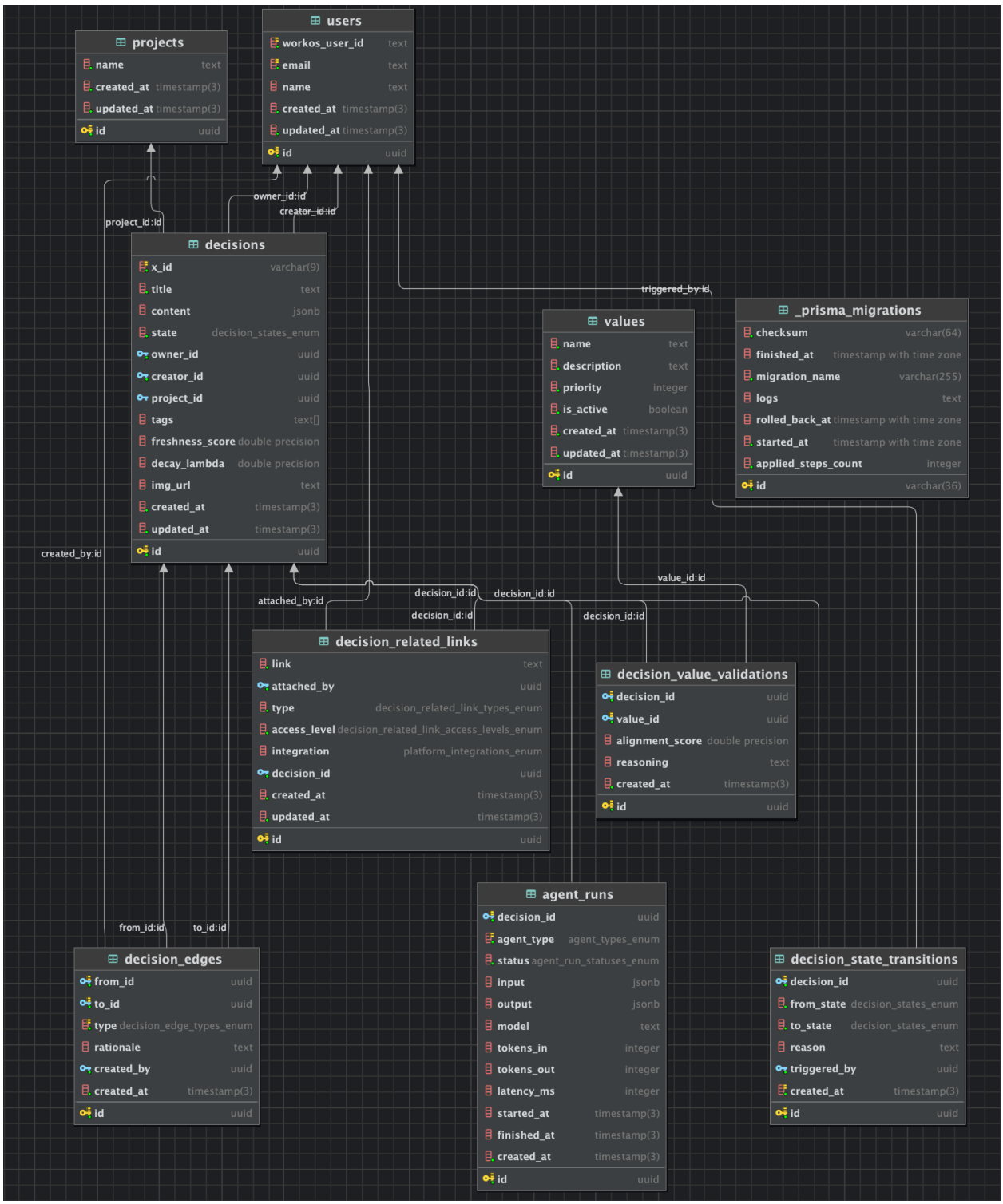

Рис. 3.2. ER-діаграма бази даних

Фрагмент схеми бази даних для FSM та DAG у форматі Prisma наведений нижче на рис. 3.3. Це безпосереднє відображення логічної моделі у ORM-нотації, що генерує TypeScript-моделі та дозволяє виконати «міграцію»

бази даних, створюючи необхідні таблиці та зв'язки. Інші моделі також наведено у Додатку А.

```
model DecisionEdge {
  id        String           @id @default(dbgenerated("gen_random_uuid()")) @db.Uuid
  fromId    String           @map("from_id") @db.Uuid
  toId      String           @map("to_id") @db.Uuid
  type      DecisionEdgeType
  rationale String?          // обґрунтування зв'язку

  from      Decision @relation("EdgeFrom", fields: [fromId], references: [id])
  to        Decision @relation("EdgeTo", fields: [toId], references: [id])
  creator   User     @relation("EdgeCreator", fields: [createdBy], references: [id])
  createdBy String   @map("created_by") @db.Uuid

  createdAt DateTime @default(now()) @map("created_at")

  @@unique([fromId, toId, type])
  @@index([fromId])
  @@index([toId])
  @@map("decision_edges")
}

// ═══ FSM transition log (підрозділ 2.1) ═══

model DecisionStateTransition {
  id         String        @id @default(dbgenerated("gen_random_uuid()")) @db.Uuid
  decisionId String        @map("decision_id") @db.Uuid
  fromState  DecisionState @map("from_state")
  toState    DecisionState @map("to_state")
  reason     String?

  decision    Decision @relation(fields: [decisionId], references: [id])
  triggeredBy String   @map("triggered_by") @db.Uuid
  user        User     @relation("TransitionTrigger", fields: [triggeredBy], references: [id])

  createdAt DateTime @default(now()) @map("created_at")

  @@index([decisionId, createdAt])
  @@map("decision_state_transitions")
}
```

Рис. 3.3. FSM & DAG – Prisma Schema

3.3 Розробка інтерфейсів програмного забезпечення

Розробка інтерфейсу інформаційної системи «DECY» зосереджена на створенні робочого середовища, в якому управлінські рішення стають

видимими, трасованими та керованими. Основна мета – зробити взаємодію з рішеннями настільки ж легкою, наскільки сьогодні є робота із задачами в трекерах на кшталт Linear чи Jira, не вимагаючи від користувача додаткового когнітивного навантаження. Відповідно до користувацького досвіду систему розділено на п’ять основних модулів, кожен з яких реалізує окремий функціонал:

– реєстр рішень;
– граф залежностей;
– картка окремого рішення;
– Decision Composer для швидкого фіксування рішень;
– AI-council модуль.

3.2.1 Реєстр рішень

Основна точка входу в систему та початкова сторінка – реєстр усіх рішень компанії з їх станом, типом, критичністю та показником актуальності (рис. 3.4).

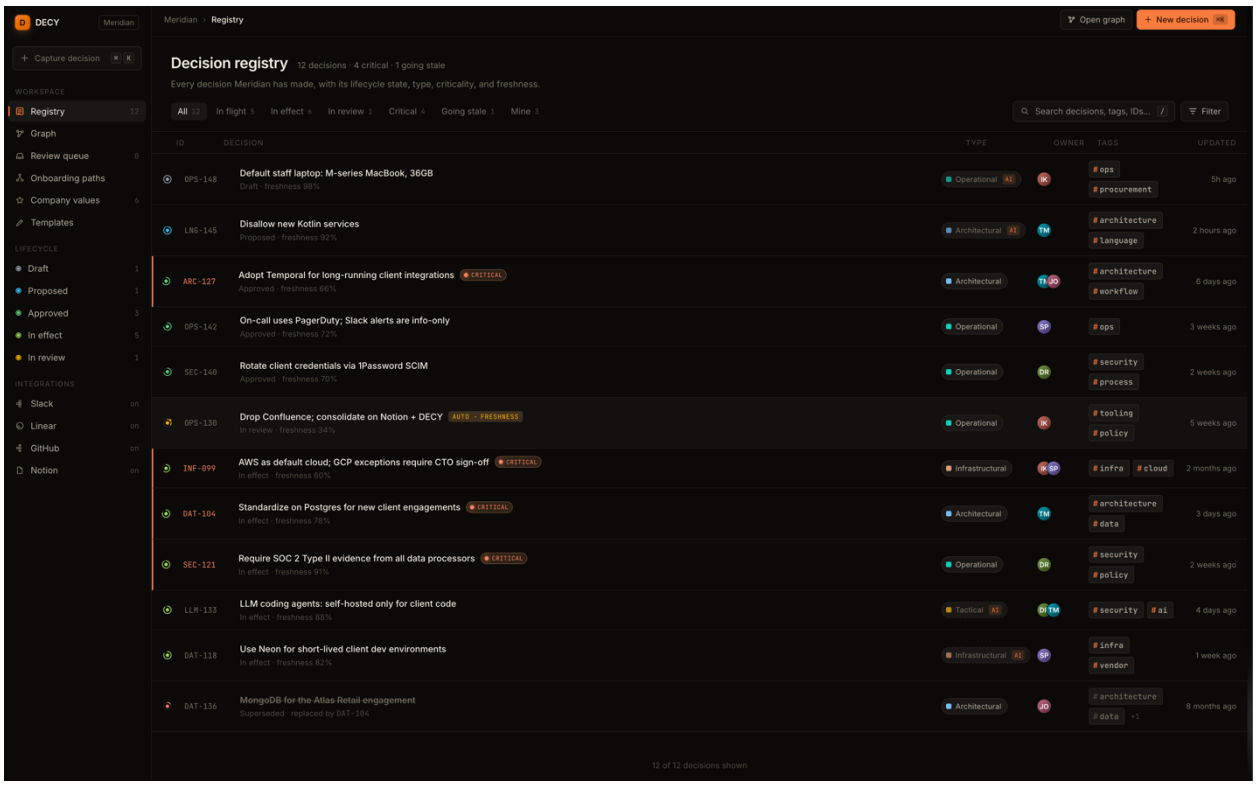

Рис. 3.4. Фрагмент інтерфейсу – реєстр рішень

На інтерфейсі реєстру рішень ліворуч розташована навігаційна панель із фільтрами за станом життєвого циклу рішень (див. розділ 2.1). Кожен рядок реєстру містить унікальний ідентифікатор у форматі XXX-NNN, де $X = [A - Z]$, $N = [0 - 9]$, заголовок рішення, поточний стан із показником *freshness*, тип рішення, власник рішення та теги. Критичні рішення підсвічуються спеціальним маркером, рішення в стані *Superseded* відображаються закресленими та з посиланням на замінник. Це базовий екран, з якого користувач переходить до будь-якого іншого розрізу системи.

3.3.2 Картка рішення

Коли користувач вибирає окреме рішення – відображається окремий екран з карткою вибраного рішення. Фрагмент інтерфейсу картки рішення зображено на рис. 3.5.

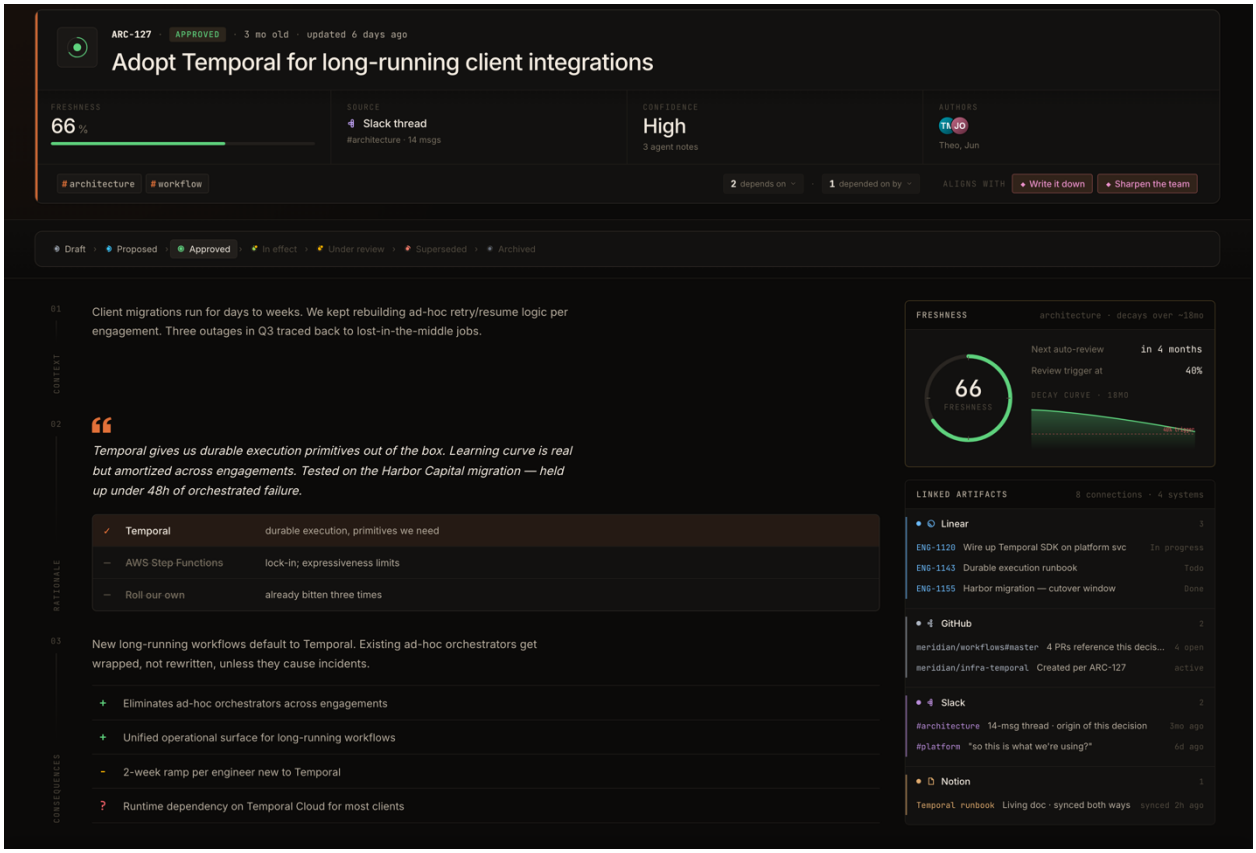

Рис. 3.5. Фрагмент інтерфейсу картки рішення

Інтерфейс картки реєстру рішень (рис. 3.7) трансформує складну бекенд-логіку у зручний інструмент для щоденної роботи команд. Верхня панель

виконує роль трекера життєвого циклу – вона є візуальною проєкцією моделі скінченного автомата (FSM) з підрозділу 2.1, що наочно підсвічує поточний статус ухваленого рішення. Також, у верхньому блоці відображено блок відповідності цінностям (Alignment), який показує ступінь узгодженості рішення зі стратегічними орієнтирами компанії, блок зв'язків з іншими рішеннями та список власників (хто відповідає за це рішення).

Центральна область екрана поділена на функціональні блоки, які розкривають суть питання: контекст обговорення, аргументоване обґрунтування з розглянутими альтернативами та прогнозовані наслідки реалізації конкретного рішення.

Особливу цінність для управління проєктом становить права бічна панель. Її верхній сегмент виступає елементом для моніторингу застарілості та необхідності його перегляду – виведено поточний показник актуальності (Freshness Score), графік експоненціальної кривої спадання, а також прогноз дати наступного автоматичного перегляду на основі заданого порогового значення. Нижче розміщено блок інтегрованих артефактів – єдину точку доступу до зовнішніх систем. Завдяки двосторонній синхронізації, система агрегує пов'язані задачі з Linear, код із GitHub, обговорення зі Slack та документи з Notion, формуючи вичерпний контекст рішення без потреби залишати поточне робоче вікно.

3.3.3 Граф залежностей

При натисканні на кнопку «Graph» на картці рішення, відкривається окремий екран з графом зв'язків між рішеннями у режимі «Focus Mode», що візуалізує DAG-модель з підрозділу 2.2. Сторінка має два режими: Focus Mode та Global View:

- Focus Mode – користувач обирає одне рішення, на якому потрібно сфокусуватись, і граф автоматично показує його найближче оточення: upstream-рішення (на яких воно базується) та downstream-рішення (які залежать від нього). Фрагмент інтерфейсу зображено на рис. 3.6;

– Global Layout – користувач бачить всю карту рішень та залежності між ними.

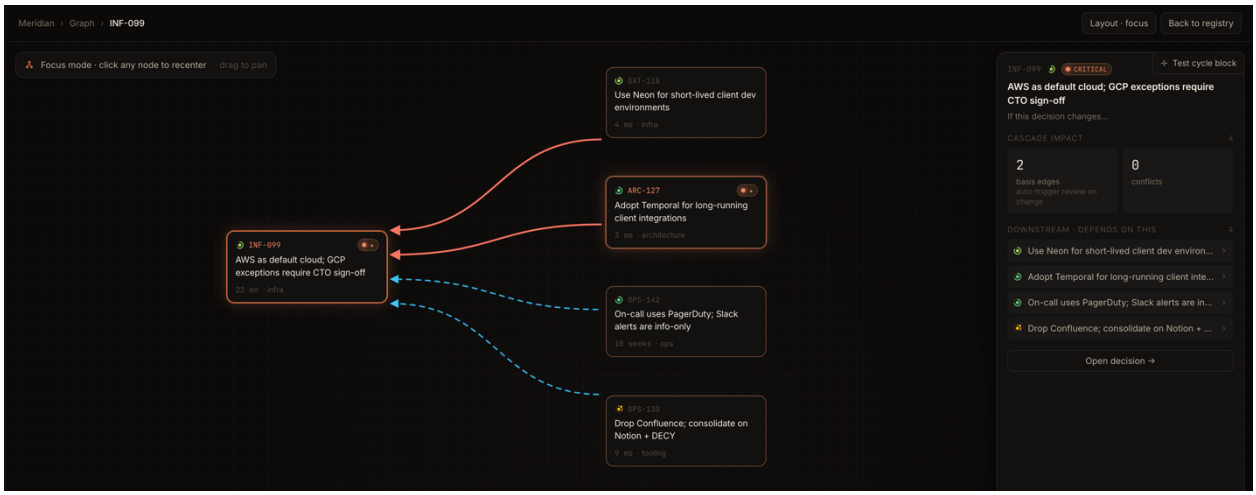

Рис. 3.6. Фрагмент інтерфейсу графової моделі (Focus Mode)

Кольорове кодування ребер відповідає типам зв’язків з підрозділу 2.2.1:

– червоні стрілки – basis-залежності з критичним каскадним впливом;
– сині пунктирні – звичайні (soft dependency) зв’язки.

Права панель показує результати каскадного аналізу – кількість basis-ребер, кількість конфліктів, перелік upstream та downstream рішень з можливістю одним кліком переключити фокус на будь-яке з них. Така подача дозволяє відстежувати наслідки потенційної зміни рішення без потреби тримати всю карту залежностей у голові.

Режим Global Layout задля виконання встановлених строків по розробці проєкту було вирішено відкласти на Post-MVP фазу.

3.3.4 Універсальний composer для фіксування рішень

Для найшвидшого фіксування рішень запропоновано використовувати універсальний composer, доступний з будь-якого екрану через гарячу клавішу *CMD + K* (рис. 3.7).

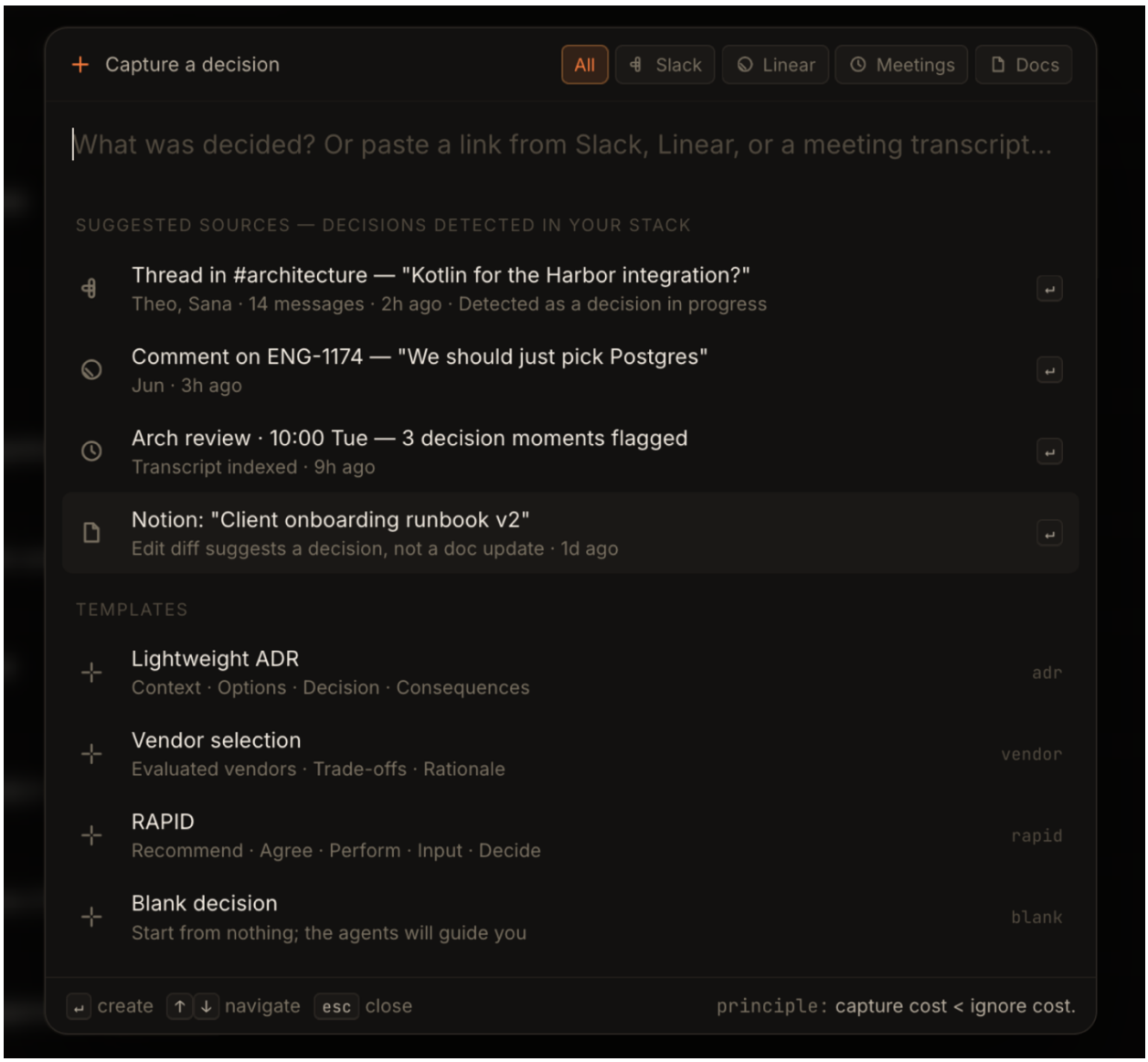

Рис. 3.7. Decision Capture

Рішення може бути створене (зафіксоване) з чотирьох джерел:

– вільний текст;
– посилання на Slack-обговорення;
– Linear-коментар;
– транскрипт зустрічі.

Над полем вводу платформа показує проактивні підказки – обговорення в інтегрованих системах, які автоматично класифіковані як «потенційні рішення в процесі». Нижче зображена бібліотека шаблонів (Lightweight ADR, Vendor selection, RAPID, чистий лист), кожен з яких призначений для

конкретного класу рішень. Вся навігація у межах composer також доступна з клавіатури.

3.3.5 AI-council модуль

Модуль AI-council (рис. 3.8) є візуальним інтерфейсом взаємодії з ШІ-системою платформи. Згідно з архітектурним задумом, інтелектуальна надбудова виконує функцію радника. Система вимагає від власників рішення явного ознайомлення та реакції на виявлені ризики, які було знайдено за допомогою ШІ-агентів.

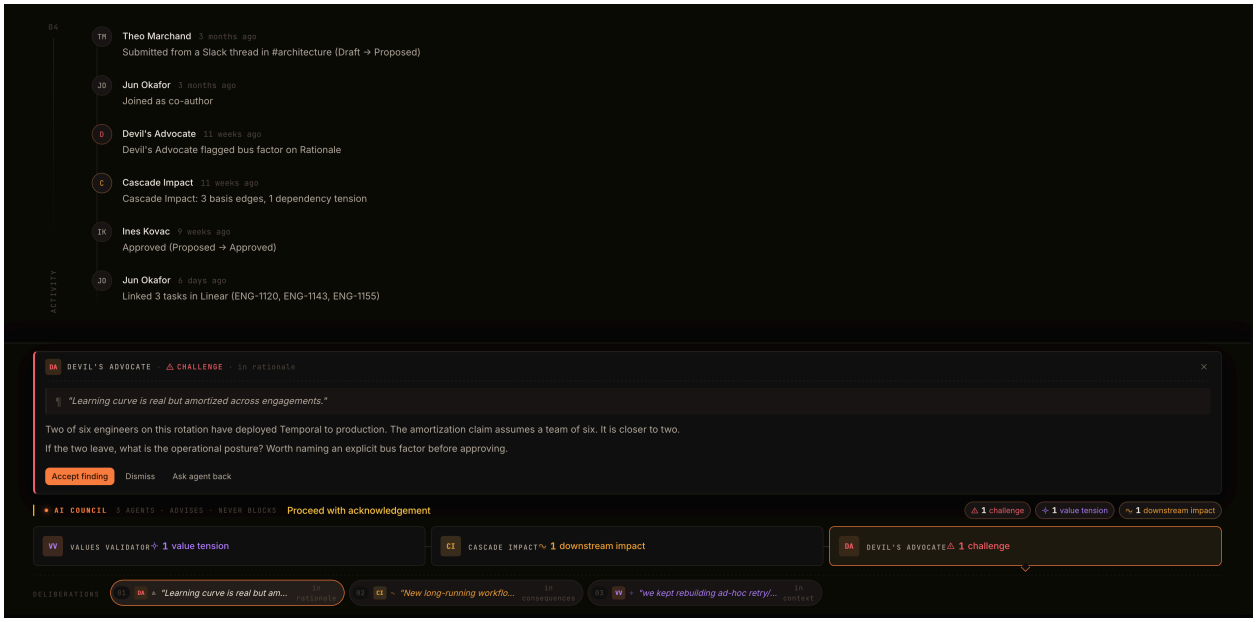

Рис. 3.8. Фрагмент інтерфейсу AI-council

На наведеному фрагменті інтерфейсу продемонстровано результати паралельної роботи трьох вузькоспеціалізованих агентів:

- ***Devil's Advocate*** – аналізує текст обґрунтування на предмет когнітивних упереджень та логічних помилок;
- ***Values Validator*** – ідентифікує потенційні конфлікти обраного варіанта з формалізованими корпоративними цінностями;
- ***Cascade Impact*** – взаємодіє з DAG-моделлю для виявлення наслідків для інших залежних рішень у системі.

Критерії оцінювання якості рекомендацій ШІ-агентів

Підхід до оцінки якості роботи агентів безпосередньо залежить від їхньої задачі. Для детермінованих алгоритмів, таких як Cascade Impact, ключовим критерієм є коректність самої математичної моделі, що перевіряється класичними модульними тестами обходу DAG (див. підрозділ 2.2.2).

Для оцінки стохастичних LLM-агентів (Devil's Advocate, Values Validator) застосовується система операційних метрик, сфокусованих на ефективності, точності та релевантності їхніх відповідей:

- ***Dismissal rate*** (частка відхилених порад) – демонструє, наскільки часто агент генерує нерелевантні знахідки. Високий показник сигналізує про необхідність допрацювати системний промпт або розширити джерела контексту;
- ***Acceptance rate*** (частка прийнятих порад) – виступає прямим індикатором короткострокової корисності агента для користувача.
- ***Ratio of consequential interactions*** – стратегічно найважливіша метрика, що показує частку порад, які призвели до фактичної зміни рішення чи його обґрунтування. Вона відображає реальний вплив ШІ на якість управління, а не просто суб’єктивну реакцію користувача.

РОЗДІЛ 4. УПРАВЛІННЯ РИЗИКАМИ ПРОЄКТУ ТА ПЛАН ПІЛОТНОГО ВПРОВАДЖЕННЯ

Фінальний етап планування проєкту «DECY» зосереджений на управлінні ризиками та підготовці до пілотного запуску. Будь-який програмний продукт доводить свою ефективність лише під час зіткнення з реальними операційними процесами компанії-замовника. Відповідно, цей розділ описує стратегію мінімізації проєктних загроз та пропонує практичний план інтеграції системи в робоче середовище ТОВ «ЕЗІК» для перевірки закладених продуктових гіпотез.

4.1 Ідентифікація ризиків проєкту та планування методів реагування

Управління ризиками проєкту «DECY» складається з чотирьох послідовних процесів:

- ідентифікація ризиків;
- якісна та кількісна оцінка;
- планування реагування;
- моніторинг та контроль.

Кожен ризик характеризується трьома параметрами:

- ймовірністю настання (P) за п’ятибальною шкалою;
- ступенем впливу на проєкт (I) за п’ятибальною шкалою;
- інтегральним показником рейтингу ризику $R = P \times I$, що визначає його пріоритет в реєстрі.

Стратегії реагування обираються з чотирьох канонічних варіантів:

- ***Avoid*** – змінити план так, щоб виключити ризик;
- ***Mitigate*** – знизити ймовірність або вплив;
- ***Transfer*** – перекласти наслідки на третю сторону;

– ***Accept*** – свідомо погодитися з можливими наслідками, формуючи резерв на реагування.

Для кожного ризику призначається власник – конкретна роль у команді, що несе оперативну відповідальність за моніторинг та виконання плану реагування.

Для забезпечення стійкості проєкту розробки системи «DECY» було проведено якісний аналіз ризиків. За його результатами сформовано реєстр (табл. 4.1) із 15 ключових загроз, які класифіковано за трьома віхами (технологічні, комерційні та організаційні) та оцінено за показниками P, I та R зазначеними вище.

Таблиця 4.1

Реєстр ризиків

№	Опис ризику	P	I	R	Стратегія реагування	Власник
1	**2**	**3**	**4**	**5**	**6**	**7**
T	**Технологічні ризики**					
T1	Концентрація архітектурної, бекенд- та DevOps-компетенцій в одній особі (точка єдиної відмови / Single point of failure)	3	5	15	Зниження (Mitigate): ведення ADR для всіх архітектурних рішень; практика парного програмування з Backend-інженером.	PM
T2	Затримка на переході M2/M3 (передача архітектури), що блокує одночасно три критичні гілки WBS	3	5	15	Зниження (Mitigate): проміжна верифікація архітектури в місяці M1.5; жорсткий контроль (gate review) у M2.	PM
T3	Недостатня якість ШІ-агентів для production	1	4	4	Прийняття (Accept): ризик приймається через високу базову зрілість обраних LLM.	AI/ML Eng.
T4	Залежність від вендора LLM-провайдерів (Anthropic, Google)	2	3	6	Зниження (Mitigate): абстракція LLM-провайдера через інтерфейси на рівні бекенду;	AI/ML Engineer

Продовження таблиці 4.1

1	2	3	4	5	6	7
T5	Накопичення технічного боргу через стиснутий 6-місячний графік розробки	4	3	12	Прийняття (Accept): свідоме фіксування боргу; виділення 1-2 спринтів в Q3 виключно на рефакторинг та оптимізацію коду.	Tech Lead
C	**Комерційні ризики**					
C1	Низька відповідність ринку (Product-Market Fit), пілотні клієнти не бачать достатньої цінності в продукті	3	5	15	Зниження (Mitigate): щотижневі глибинні інтерв'ю під час пілоту; готовність до стратегічного розвороту візії продукту.	PM
C2	Тривалий цикл продажу B2B SaaS (3–6 міс. на клієнта) сповільнює динаміку зростання	4	3	12	Зниження (Mitigate): продаж через прямі рекомендації від ТОВ «ЕЗІК»; фокус на закриті IT-спільноти як основний канал збуту.	PM
C3	Вихід прямих конкурентів (Cloverpop, Bizzdesign) на ринок SMB з аналогічним функціоналом	2	4	8	Зниження (Mitigate): забезпечення переваги першого гравця (first-mover); вузьке фокусування виключно на IT-вертикалі.	PM
C4	Недостатня воронка лідів через відсутність значного маркетингового бюджету на ранньому етапі	4	3	12	Зниження (Mitigate): контент-маркетинг (експертні статті, демо); модель Product-led growth через базовий безкоштовний тариф.	PM
C5	Валютні коливання (клієнти-резиденти України сплачують у гривні, тоді як інфраструктурні витрати в USD)	2	2	4	Прийняття (Accept): фіксація курсу в SLA на момент укладення контракту; перегляд умов не частіше одного разу на рік.	PM
O	**Організаційні ризики**					
O1	Втрата ключового члена команди (Tech Lead, AI/ML) в період активної розробки	2	5	10	Зниження (Mitigate): бонуси за утримання для критичних ролей; підтримка документації	PM

Продовження таблиці 4.1

1	2	3	4	5	6	7
O2	Форс-мажорні геополітичні фактори (повітряні тривоги, відключення електроенергії), що впливають на продуктивність	4	3	12	Зниження (Mitigate): забезпечення офісу автономним живленням; надання команді можливості віддаленої роботи або релокації.	PM
O3	Розфокусування замовника (ТОВ «ЕЗІК») між основним операційним консалтингом та розробкою продукту «DECY»	3	4	12	Зниження (Mitigate): формальне виділення проєктної команди у незалежну одиницю; жорсткий розподіл робочого часу.	PM
O4	Неадекватність обраних KPI пілоту, складність вимірювання реального впливу системи на якість рішень	3	4	12	Зниження (Mitigate): фіксація базових метрик до старту проєкту; використання контрольної групи під час пілоту.	PM
O5	Юридичні колізії при передачі інтелектуальної власності від членів команди до ТОВ «ЕЗІК»	2	4	8	Зниження (Mitigate): підписання GIG-контрактів на базі рекомендованого шаблону Дія Сіті із чіткою передачею майнових прав (IP-assignment)	PM

Для візуалізації пріоритезації ризиків побудовано матрицю probability × impact, де кожен ризик розміщений у відповідній клітинці згідно з його оцінкою P та I (рис. 4.1). Зони матриці позначені за рівнем критичності:

- червона зона ($R \geq 12$) – ризики високого пріоритету, що вимагають активного управління та виділеного власника;
- жовта зона ($6 \leq R < 12$) – ризики середнього пріоритету, що потребують моніторингу;
- зелена зона ($R < 6$) — низькоприоритетні ризики, що приймаються або моніторяться пасивно.

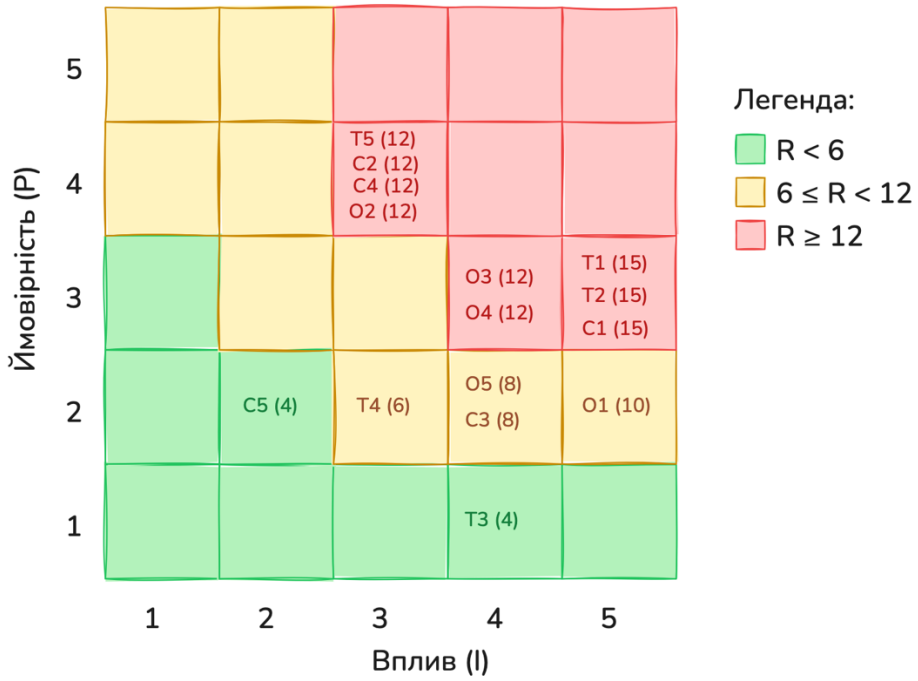

Рис. 4.1. Матриця ймовірностей та впливу ризиків

4.2 Планування пілотного впровадження системи «DECY»

Пілотне впровадження є завершальним етапом проєктного циклу та точкою фактичного зіткнення продукту з операційною реальністю. ТОВ «ЕЗІК» обрано першим замовником пілоту з трьох причин:

– компанія є водночас розробником і користувачем системи, що дає доступ до зворотного зв’язку та можливість швидкої валідації гіпотез;

– організаційна структура «ЕЗІК» (не більше 10 співробітників, IT-консалтинг та продуктова розробка) точно відповідає цільовому сегменту IT-SME;

– внутрішнє впровадження продукту, який сам ще розробляється, втілює принцип *dogfooding* – команда розробників та проєктних керівників щодня користуються власним продуктом, що різко прискорює виявлення UX-проблем та функціональних проблем.

4.2.1 Базові метрики до старту пілоту (baseline)

Кількісна оцінка ефекту впровадження неможлива без фіксації стартового стану процесів прийняття рішень. Перед початком пілоту в ТОВ «ЕЗІК» проведено вимірювання шести ключових метрик, що формують точку відліку для подальшого порівняння (табл. 4.2). Метрики обрано так, щоб одночасно охоплювати кількісні та якісні аспекти процесу прийняття рішень.

Таблиця 4.2

Базові метрики ТОВ «ЕЗІК» до старту пілоту

Метрика	Baseline	Метод вимірювання
M1. Кількість управлінських рішень за місяць	~60 рішень	Аудит Slack-каналів, Jira-коментарів, нотаток з мітингів за репрезентативний місяць
M2. Decision capture rate (частка формально задокументованих рішень)	~30%	Перетин «згаданих у обговореннях» та «зафіксованих»
M3. Time-to-decision (медіана часу від обговорення до рішення)	5 робочих днів	Час від першої згадки до підтвердження та початку реалізації
M4. Decision freshness (частка активних рішень, що переглянуті за останні 6 міс.)	~25%	Аналіз дат останньої модифікації для діючих задокументованих рішень
M5. Founder bottleneck index (частка рішень, що потребували підтвердження зі сторони CEO)	~70%	Аналіз всіх рішень та інтерв'ю з CEO
M6. Integration coverage (% рішень, пов'язаних з артефактами в Jira/GitHub/Confluence)	~15%	Аудит ручних посилань між системами

4.2.2 Фази пілотного впровадження

План пілотного впровадження (табл. 4.3) розбитий на декілька фаз для його поетапної реалізації.

Таблиця 4.3

Фази пілотного впровадження «DECY» в ТОВ «ЕЗІК»

Фаза	Тривалість	Цілі фази	Definition of Done
F1. Onboarding	M1	Налаштування workspace для ЕЗІК; адаптація шаблону життєвого циклу; заповнення бази корпоративних цінностей; підключення інтеграцій; тренінг команди (2 воркшопи)	100% активних користувачів; >= 5 пілотних рішень створено
F2. Active use	M2	Перенесення поточних активних рішень з legacy-каналів у DECY; усі нові рішення створюються через DECY; запуск ШІ-агентів	>50 рішень в системі; <=80% нових рішень фіксуються через DECY; >=30 корисних спрацьовувань агентів
F3. Validation & expansion	M3	Інтеграція DECY у щотижневий ритм компанії; кількісне порівняння за всіма 6 метриками; формування внутрішнього case study	Звіт з порівнянням метрик до/після; >= 3 внутрішні рецензії готові для зовнішніх клієнтів

Принципова особливість фази F1 на старті пілоту полягає у тому, що команда отримує саме налаштовану систему. Шаблон життєвого циклу попередньо адаптовано під операційні особливості ЕЗІК, базу цінностей наповнено за результатами окремої сесії з засновником, інтеграції налаштовані до офіційного старту. Це знижує бар’єр входу команди в систему.

Фаза F2 переводить компанію у режим повної заміни попередніх каналів фіксування рішень. Команда отримує жорстке правило: рішення не існує, поки воно не зафіксоване в «DECY».

Фаза F3 найбільш аналітично навантажена. На цьому етапі система вже накопичила достатньо даних для статистично значущого порівняння з базовими метриками, і керівник проєкту разом із засновником ЕЗІК виконують вимірювання за тими ж шістьма метриками з табл. 4.2.

Типові сценарії використання «DECY» у рамках пілоту

Окрім кількісних метрик, успіх пілоту визначається повнотою покриття ключових функцій інформаційної системи реальною роботою користувачів. Для забезпечення такого покриття сформовано перелік п’яти типових сценаріїв використання, що тестуються командою ТОВ «ЕЗІК» протягом пілоту (табл. 4.4).

Таблиця 4.4

Сценарії використання у рамках пілоту

№	Сценарій	Користувач	Фаза	Послідовність кроків
1	2	3	4	5
S1	Створення та фіксація нового рішення з повним проходженням FSM-циклу	PM, Tech Lead	F1, F2	Користувач ініціює рішення через universal composer → заповнює context, alternatives, rationale → подія submit (Draft → Proposed) → агенти Devil's Advocate та Values Validator формують знахідки → користувач реагує (accept / dismiss / ask back) → approve (Proposed → Approved) → activate (Approved → InEffect)
S2	Імпорт рішення з зовнішніх каналів та прив'язка до DAG	PM, Tech Lead	F2	Користувач створює запис рішення на основі обговорення зі Slack або Confluence → заповнює мінімальний контекст з посиланням на оригінальне джерело → додає ребра basis до раніше зафіксованих рішень → Cascade Impact верифікує цілісність графа залежностей
S3	Каскадний аналіз впливу перед зміною критичного рішення	Tech Lead, CTO	F2, F3	Користувач відкриває рішення з високим показником PageRank → переглядає множину у візуалізації графа → оцінює обсяг каскадного перегляду → ініціює подію trigger_review для всіх залежних рішень або відкладає зміну.
S4	Автоматичний trigger_review застарілого рішення через спадання freshness	Власник рішення	F2, F3	Система виявляє рішення з freshness_score нижче порогу → автоматично переводить його в стан UnderReview → надсилає сповіщення власнику → власник переглядає та ухвалює подальшу дію.

Продовження таблиці 4.4

1	2	3	4	5
S5	Онбординг нового співробітника	Новий співробітник та PM	F3	Новий співробітник отримує доступ до workspace → система генерує персоналізований маршрут з топ-10 рішень, відсортованих за PageRank → користувач послідовно ознайомлюється з фундаментальними архітектурними та процесними рішеннями → за результатами формує запитання до власників рішень через механізм коментарів

4.2.3 Критерій успіху пілоту

Загальний успіх пілоту визначається досягненням SMART-критеріїв за кожною з базових метрик (табл. 4.5). Цільові значення встановлено консервативно. Пілот вважається успішним, якщо досягнуто щонайменше чотирьох критеріїв з шести, оскільки повне досягнення всіх KPI на першому пілоті очікується малоймовірним і свідчило б скоріше про надто легкі цілі, ніж про реальний успіх.

Таблиця 4.5

Цільові показники успіху пілоту

Метрика	Baseline	Ціль після пілоту	Дельта
M1. Кількість фіксованих рішень/міс.	~60	~90	+50%
M2. Decision capture rate	30%	75%	+45 п.п.
M3. Time-to-decision (медіана), днів	9	5	-44%
M4. Decision freshness	25%	60%	+35 п.п.
M5. Founder bottleneck index	70%	45%	-25 п.п.
M6. Integration coverage	15%	70%	+55 п.п.

4.3 Стратегія масштабування продукту на зовнішніх клієнтів ТОВ «ЕЗІК»

Успішне завершення внутрішнього пілоту в ТОВ «ЕЗІК» є необхідним, але не кінцевим етапом розвитку продукту. Стратегія подальшого масштабування платформи «DECY» на зовнішній B2B-ринок складається з трьох послідовних кроків:

- ***Формування доказової бази (Case Study).*** На основі результатів внутрішнього пілоту створюється детальний звіт. Він міститиме кількісне порівняння стартових метрик із результатами після впровадження, опис реальних сценаріїв використання та відгуки команди. Цей документ стане основним інструментом для залучення клієнтів методом *cold outreach*.
- ***Зовнішнє пілотне впровадження.*** Через професійну мережу контактів засновника планується залучити 3–5 зовнішніх компаній (українські та європейські IT-підприємства сегмента SME з командою 5–30 співробітників). На цьому етапі застосовується модель безкоштовного пілоту строком на 3 місяці з подальшою конверсією у платний тариф після підтвердження цінності системи.
- ***Комерційний запуск.*** Затвердження або перегляд цінової моделі та повноцінний вихід на ринок. Жорстким критерієм переходу до цього кроку є завершення щонайменше двох успішних зовнішніх пілотів з високим показником індекса лояльності клієнтів NPS .

4.3.1 Ідеальний профіль клієнта

Для ефективної реалізації другого та третього кроків масштабування сформовано ідеальний профіль клієнта (рис. 4.2).

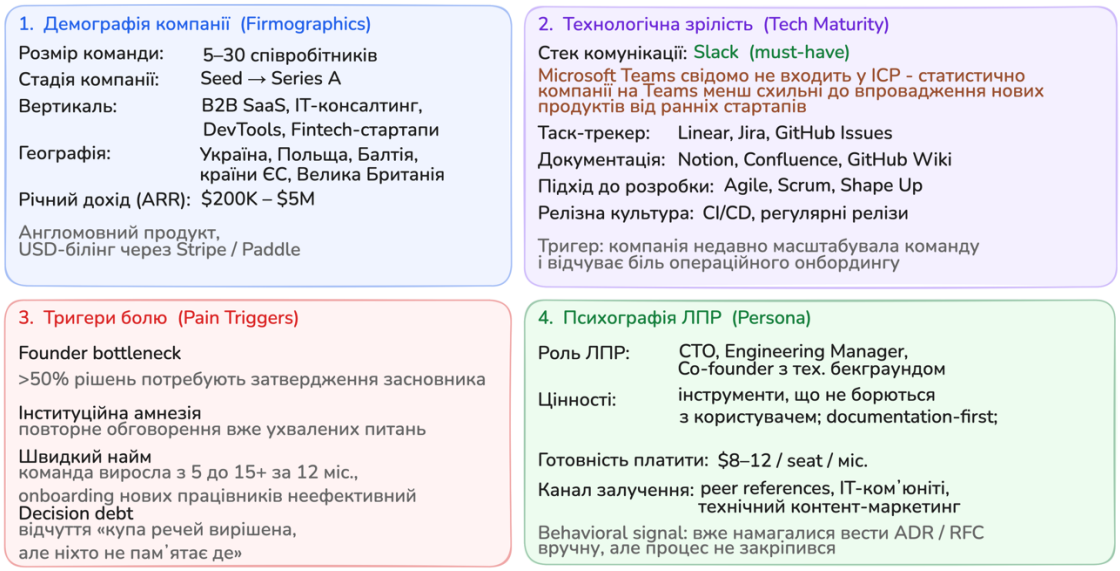

Рис. 4.2. Ідеальний профіль клієнта

Продукт «DECY» не має на меті охопити весь ринок, а фокусується виключно на IT-компаніях малого та середнього бізнесу. Характерними рисами цільової компанії є наявність розподіленої або гібридної команди, використання сучасного стеку інструментів (Slack, Jira/Linear, Notion) та наявність проблем з управлінням рішеннями.

4.3.2 Дорожня карта (roadmap) розвитку «DECY»

Загальний часовий горизонт реалізації стратегії масштабування від початку внутрішнього тестування до комерційного релізу оцінюється у 9 місяців. Така динаміка (табл. 4.6) повністю відповідає типовому життєвому циклу посівної стадії (seed stage) технологічного IT-стартапу.

Таблиця 4.6

Roadmap розвитку «DECY»

Етап	Місяці	Ключові активності	Очікуваний результат
1	**2**	**3**	**4**
Внутрішній пілот	M1–M3	Розгортання системи в ТОВ «ЕЗІК», навчання команди, калібрування ШІ-агентів, збір метрик	Досягнення цільових показників успіху (KPI) пілоту

Продовження таблиці 4.6

1	2	3	4
Пакування цінності	M4	Аналіз результатів, написання Case Study, підготовка презентаційних матеріалів	Готовий пакет маркетингових матеріалів для B2B-продажів
Зовнішні пілоти	M5–M7	Залучення 3–5 IT-компаній, онбординг зовнішніх команд, збір зворотного зв'язку	>= 2 успішних зовнішніх впроваджень з NPS >= 40
Комерційний реліз	M8–M9	Налаштування білінгу, підготовка інфраструктури до навантажень, старт публічних рекламних кампаній	Перші комерційні контракти (MRR > 0)

4.3.3 Трансформація проєктної команди «DECY» на етапі зовнішнього впровадження

Вихід продукту на зовнішній ринок та перехід до етапу комерційного релізу вимагають відповідної трансформації проєктної команди ТОВ «ЕЗІК». Якщо на етапах розробки та внутрішнього пілоту (M1–M3) основний ресурс був сфокусований на розробці та створенні функціоналу *(capex)*, то починаючи з п’ятого місяця фокус поступово зміщується на операційну підтримку *(opex)*.

У контексті IT-проєктів зміна фінансової моделі означає перехід між двома категоріями витрат:

- ***CAPEX (Капітальні витрати)*** – інвестиції у створення нематеріального активу – інформаційної системи «DECY», її архітектури та ШІ-моделей, тощо. Це стратегічні ресурси, витрачені на етапі розробки для формування майбутньої цінності продукту;
- ***OPEX (Операційні витрати)*** – регулярні витрати, необхідні для щоденної підтримки та масштабування готової системи.

Після переходу до *opex*, проєктний менеджер може частково взяти на себе ролі Customer Success Manager, забезпечуючи успішний онбординг перших зовнішніх клієнтів. Технічна команда, у свою чергу, переходить від

моделі активної розробки нового функціоналу до моделі забезпечення надійності (SRE) та клієнтської підтримки другої лінії. З точки зору методології управління, цей перехід є класичним етапом закриття проєкту розробки та передачі системи в стабільну операційну діяльність.

ВИСНОВКИ

У даній кваліфікаційній роботі магістра вирішено актуальну науково-практичну задачу – розроблено концепцію інформаційної системи підтримки прийняття управлінських рішень «DECY» для ринку Decision Intelligence та сформовано план управління проєктом створення і впровадження інформаційної системи.

Особливістю вирішеної задачі є її мета-проблематика (подвійний фокус). Об’єктом розробки виступає ІТ-продукт, який сам по собі є спеціалізованим інструментом проєктного менеджменту, безпосередньо спрямованим на розв’язання фундаментальних проблем управління ІТ-проєктами.

Проведене дослідження дозволило зробити наступні теоретичні та практичні висновки:

- На основі системного аналізу предметної області встановлено, що сучасні ІТ-компанії малого та середнього бізнесу стикаються з критичною проблемою неформалізованості управлінських процесів та рішень. Це призводить до накопичення «боргу рішень», втрати бізнес-контексту, та виникнення ефекту «founder bottleneck» (залежності швидкості розробки від мікроменеджменту засновників). Доведено, що для подолання цих викликів необхідний перехід від розрізнених та нестандартизованих каналів фіксації рішень до єдиної спеціалізованої інформаційної системи.
- Проведено маркетингове дослідження B2B SaaS-ринку Decision Intelligence та стратегічний і конкурентний аналіз проєкту. За допомогою побудови дерева цілей та проблем, а також проведення SWOT-аналізу, обґрунтовано стратегічне позиціонування інформаційної системи «DECY». Встановлено, що розроблювана система виступає не лише ІТ-продуктом, але й спеціалізованим

інструментом для методології управління проєктами, забезпечуючи формалізацію та збереження управлінського контексту організації.

- Розроблено математичні моделі, що склали алгоритмічне ядро системи. Застосування апарату скінченних автоматів (FSM) дозволило формалізувати життєвий цикл управлінського рішення, перетворивши його на контрольовану сутність зі статусною моделлю та функцією спадання актуальності у часі. Використання моделі спрямованого ациклічного графу (DAG) забезпечило можливість автоматизованого відстеження причинно-наслідкових зв'язків, що дозволяє керівникам прогнозувати каскадний вплив змін в IT-проєктах.
- Сформовано план управління проєктом розробки. Побудовано ієрархічну структуру робіт (WBS), розроблено календарний план з визначенням критичного шляху, запропоновано організаційну структуру проєктної команди з розподілом ролей і розраховано кошторис IT-проєкту. Для забезпечення моніторингу та контролю за виконанням спроєктовано систему показників на основі методу освоєного обсягу. Це створило надійне управлінське підґрунтя для успішної та своєчасної реалізації розробки в межах визначених ресурсних та часових обмежень.
- Здійснено проєктування технічної архітектури інформаційної системи. Обрано архітектурний патерн «модульний моноліт», розроблено логічну модель реляційної бази даних та ORM-специфікації. Спроєктовано інтерфейси користувача інформаційної системи. Особливістю технічного рішення є проєктування мультиагентної системи штучного інтелекту «AI-council» на базі фреймворку LangGraph, яка автоматизує валідацію рішень на відповідність корпоративним стандартам та виявляє когнітивні упередження в процесі планування.
- Сформовано план управління ризиками та розроблено стратегії реагування на ключові проєктні загрози. Сплановано поетапне пілотне впровадження платформи «DECY» в операційні процеси ТОВ «ЕЗІК».

Визначено ключові метрики (KPI) успішності пілоту, зокрема очікується зростання показника фіксації рішень на 45 процентних пунктів та зменшення часу на їх прийняття на 44%.

– Розроблено покрокову стратегію подальшого комерційного масштабування продукту на зовнішніх клієнтів (сегмент IT-SME). Доведено необхідність організаційної трансформації проєктної команди: перехід від капітальних витрат (CAPEX) розробки до операційних витрат (OPEX) підтримки. Побудована дорожня карта розгортання є класичним етапом закриття проєкту розробки (Project Closure) та переведенням його в стадію операційної SaaS-діяльності.

Результати кваліфікаційної роботи повністю підтверджують досягнення поставленої мети. Розроблена концепція інформаційної системи «DECY», математичні моделі та сплановані процеси управління проєктом мають високу практичну цінність, а результати роботи повністю підготовлені до впровадження у виробничу діяльність ТОВ «ЕЗІК» з перспективою подальшої комерціалізації.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. AlphaBOLD. LangGraph Agents in Production: Architecture, Costs & Real-World Outcomes. 2026. URL: https://www.alphabold.com/langgraph-agents-in-production/ (дата звернення: 05.05.2026).
2. Archive360. CapEx vs. OpEx: Revisiting On-Premises vs. Cloud Computing. URL: https://www.archive360.com/blog/revisited-capex-versus-opex-on-premises-versus-the-cloud (дата звернення: 05.05.2026).
3. Bloomfire. Why Data Decay Puts Your AI Strategy at Risk. URL: https://bloomfire.com/blog/data-decay-impact-on-ai-strategy/ (дата звернення: 05.05.2026).
4. Catalect. Building a Corporate Memory in 2026: How AI Agents Prevent Institutional Amnesia in the Enterprise. 2026. URL: https://www.catalect.io/blog/building-a-corporate-memory-in-2026-how-ai-agents-prevent-institutional-amnesia-in-the-enterprise (дата звернення: 05.05.2026).
5. Databricks. What is Directed Acyclic Graph (DAG)? URL: https://www.databricks.com/blog/what-is-dag (дата звернення: 05.05.2026).
6. IBM. Comparing AI agent frameworks: CrewAI, LangGraph, and BeeAI. URL: https://developer.ibm.com/articles/awb-comparing-ai-agent-frameworks-crewai-langgraph-and-beeai/ (дата звернення: 05.05.2026).
7. Java Code Geeks. Microservices vs Monoliths in 2026: When Each Architecture Wins. 2025. URL: https://www.javacodegeeks.com/2025/12/microservices-vs-monoliths-in-2026-when-each-architecture-wins.html (дата звернення: 05.05.2026).
8. KPMG International. Risk Modernization | AI is revolutionizing risk management. 2025. URL: https://kpmg.com/us/en/articles/2025/ai-revolutionizing-risk-management.html (дата звернення: 05.05.2026).
9. LLC “EZIC”. AI Manifesto: What We Believe – six principles that define how we build. 2026. URL: https://ezic.io/ai-manifesto (дата звернення: 05.05.2026).

10. McKinsey & Company. Decision making in the age of urgency. 2019. URL: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/Organization/Our%20Insights/Decision%20making%20in%20the%20age%20of%20urgency/Decision-making-in-the-age-of-urgency.pdf (дата звернення: 05.05.2026).

11. Miro. WBS for Software Development Guide. URL: https://miro.com/project-management/how-to-use-wbs-for-software-development/ (дата звернення: 05.05.2026).

12. Mixpanel. Product-led growth in 2026: A complete guide (and the metrics that actually matter). 2026. URL: https://mixpanel.com/blog/product-led-growth/ (дата звернення: 05.05.2026).

13. National Defense Industrial Association (NDIA). An Industry Practice Guide for Integrating Agile and Earned Value Management on Programs. 2025. URL: https://www.ndia.org/-/media/sites/ndia/divisions/ipmd/2025/ndia_ipmd_agileandevmguide_version_15_september252025final.pdf (дата звернення: 05.05.2026).

14. Neon Docs. Multitenancy with Neon. URL: https://neon.com/docs/guides/multitenancy (дата звернення: 05.05.2026).

15. NIST. AI Risk Management Framework. URL: https://www.nist.gov/itl/ai-risk-management-framework (дата звернення: 05.05.2026).

16. OpenReview. Codified Finite-state Machines for Role-playing. URL: https://openreview.net/forum?id=xSuDJTQ3Ew (дата звернення: 05.05.2026).

17. Paxcom / Forrester. Bridging the Gap Between Data and Action: Role of Decision Intelligence on Business Growth. 2024. URL: https://paxcom.ai/wp-content/uploads/2024/11/Ebook-Bridging-the-Gap-Between-Data-and-Action-Role-of-Decision-Intelligence-on-Business-Growth.pdf (дата звернення: 05.05.2026).

18. Phillips Consulting. If It Is Not Documented, It Does Not Exist: Why Every Business Must Build a Culture of Documentation. 2024. URL:

https://phillipsconsulting.net/articles_post/if-it-is-not-documented-it-does-not-exist-why-every-business-must-build-a-culture-of-documentation/ (дата звернення: 05.05.2026).

19. Port.io. What is C4 Model? Applications & Best Practices. URL: https://www.port.io/glossary/c4-model (дата звернення: 05.05.2026).

20. Project Management Institute (PMI). Earned Value Management (EVM) – Understand Agile Project Progress. URL: https://www.pmi.org/learning/library/earned-value-management-understand-agile-6567 (дата звернення: 05.05.2026).

21. React Flow. Performance. URL: https://reactflow.dev/learn/advanced-use/performance (дата звернення: 05.05.2026).

22. Saliba R. Decision Debt: The Cost of Continuing Without Clarity. Medium, 2026. URL: https://medium.com/@robert.saliba.agile.coach/decision-debt-the-cost-of-continuing-without-clarity-f0ad03902da6 (дата звернення: 05.05.2026).

23. Teamazing. Knowledge Silos and Institutional Knowledge Loss. 2025. URL: https://www.teamazing.com/blog/knowledge-silos-institutional-knowledge-loss/ (дата звернення: 05.05.2026).

24. TechnologyOne. Capex vs Opex in an as-a-Service world. URL: https://technology1.com/__data/assets/pdf_file/0006/147912/Capex-vs-Opex-in-an-as-a-service-world.pdf (дата звернення: 05.05.2026).

25. Wondrasek J. A. What the Research Actually Shows About AI Coding Assistant Productivity. 2026. URL: https://www.softwareseni.com/what-the-research-actually-shows-about-ai-coding-assistant-productivity/ (дата звернення: 05.05.2026).

26. Zinoviev I. Causal Inference in Decision Intelligence — Part 4: Directed Acyclic Graph. Medium. URL: https://medium.com/@ievgen.zinoviev/causal-inference-in-decision-intelligence-part-4-directed-acyclic-graph-52235071e0fd (дата звернення: 05.05.2026).

ДОДАТОК А

```
// ═══ User & Project context ═══

model User {
  id           String   @id @default(dbgenerated("gen_random_uuid()")) @db.Uuid
  workosUserId String   @unique @map("workos_user_id")
  email        String   @unique
  name         String?
  createdAt    DateTime @default(now()) @map("created_at")
  updatedAt    DateTime @default(now()) @updatedAt @map("updated_at")

  ownedDecisions     Decision[]               @relation("DecisionOwner")
  createdDecisions   Decision[]               @relation("DecisionCreator")
  attachedLinks DecisionRelatedLink[] @relation("LinkAttacher")
  createdEdges       DecisionEdge[]           @relation("EdgeCreator")
  triggeredTransitions DecisionStateTransition[] @relation("TransitionTrigger")

  @@map("users")
}

model Project {
  id        String     @id @default(dbgenerated("gen_random_uuid()")) @db.Uuid
  name      String
  decisions Decision[]
  createdAt DateTime   @default(now()) @map("created_at")
  updatedAt DateTime   @default(now()) @updatedAt @map("updated_at")

  @@map("projects")
}
```

Рис. А.1. User & Project Context – Prisma Schema

```
// ═══ Decision: central entity ═══

model Decision {
  id      String        @id @default(dbgenerated("gen_random_uuid()")) @db.Uuid
  xId     String        @unique @map("x_id") @db.VarChar(9) // LL-DDD-LL
  title   String
  content Json?         @db.JsonB
  state   DecisionState @default(DRAFT)

  // ownership
  owner     User   @relation("DecisionOwner", fields: [ownerId], references: [id])
  ownerId   String @map("owner_id") @db.Uuid
  creator   User   @relation("DecisionCreator", fields: [creatorId], references: [id])
  creatorId String @map("creator_id") @db.Uuid

  // project context
  project   Project? @relation(fields: [projectId], references: [id])
  projectId String?  @map("project_id") @db.Uuid

  // metadata
  tags           String[] @default([])
  freshnessScore Float?   @map("freshness_score") // 0..1, обчислюється з f(t) = e^(-λ(t-t₀))
  decayLambda    Float?   @map("decay_lambda")    // λ для конкретного рішення (підрозділ 2.1.3)
  imgUrl         String?  @map("img_url")

  // related artifacts & graph
  relatedLinks  DecisionRelatedLink[]
  outgoingEdges DecisionEdge[]            @relation("EdgeFrom")
  incomingEdges DecisionEdge[]            @relation("EdgeTo")
  transitions   DecisionStateTransition[]
  validations   DecisionValueValidation[]
  agentRuns     AgentRun[]

  createdAt DateTime @default(now()) @map("created_at")
  updatedAt DateTime @default(now()) @updatedAt @map("updated_at")

  @@map("decisions")
}
```

Рис. А.2. Decision Entity – Prisma Schema